Aalto University
School of Chemical
Engineering

Sanna Hyvönen

SEMI-AUTOMATIC PARAMETRIZATION OF DYNAMIC MODELS USING PLANT DATA

Master´s Programme in Chemical, Biochemical and Materials Engineering

Major in Chemical Engineering

Master's thesis for the degree of Master of Science in Technology submitted for inspection, Espoo, 31 August, 2017.

Supervisor **Professor Sirkka-Liisa Jämsä-Jounela**

Instructor **M.Sc. Teemu Liikala**

M.Sc. Ville Saarela

Aalto University
School of Chemical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author Sanna Hyvönen

Title of thesis Semi-automatic parametrization of dynamic models using plant data

Degree Programme Master's Programme in Chemical, Biochemical and Materials Engineering

Major Chemical Engineering

Thesis supervisor Professor Sirkka-Liisa Jämsä-Jounela

Thesis advisors M. Sc. Teemu Liikala, M. Sc. Ville Saarela

Date 31.08.2017 **Number of pages** 69 **Language** English

Abstract

The aim of this thesis was to develop a new methodology for estimating parameters of NAPCON ProsDS dynamic simulator models to better represent data containing several operating points. Before this thesis, no known methodology had existed for combining operating point identification with parameter estimation of NAPCON ProsDS simulator models.

The methodology was designed by assessing and selecting suitable methods for operating space partitioning, parameter estimation and parameter scheduling. Previously implemented clustering algorithms were utilized for the operating space partition. Parameter estimation was implemented as a new tool in the NAPCON ProsDS dynamic simulator and iterative parameter estimation methods were applied. Finally, lookup tables were applied for tuning the model parameters according to the state.

The methodology was tested by tuning a heat exchanger model to several operating points based on plant process data. The results indicated that the developed methodology was able to tune the simulator model to better represent several operating states. However, more testing with different models is required to verify general applicability of the methodology.

Keywords Data mining, parameter estimation, cluster analysis, dynamic simulation

Tekijä Sanna Hyvönen

Työn nimi Dynaamisten mallien puoliautomaattinen parametrisointi käyttäen laitosdataa

Koulutusohjelma Master's Programme in Chemical, Biochemical and Materials Engineering

Pääaine Chemical Engineering

Työn valvoja Professori Sirkka-Liisa Jämsä-Jounela

Työn ohjaajat DI Teemu Liikala, DI Ville Saarela

Päivämäärä 31.08.2017 **Sivumäärä** 69 **Kieli** Englanti

Tiivistelmä

Tämän diplomityön tarkoitus oli kehittää uusi parametrien estimointimenetelmä NAPCON ProsDS -simulaattorin dynaamisille malleille, jotta ne vastaisivat paremmin dataa useista prosessitiloista. Ennen tätä diplomityötä NAPCON ProsDS -simulaattorin malleille ei ollut olemassa olevaa viritysmenetelmää, joka yhdistäisi operointitilojen tunnistuksen parametrien estimointiin.

Menetelmän kehitystä varten tutkittiin ja valittiin sopivat menetelmät operointiavaruuden jakamiselle, parametrien estimoinnille ja parametrien virittämiseen prosessitilan mukaisesti. Aikaisemmin ohjelmoituja klusterointialgoritmeja hyödynnettiin operointiavaruuden jakamisessa. Parametrien estimointi toteutettiin uutena työkaluna NAPCON ProsDS -simulaattoriin ja estimoinnissa käytettiin iteratiivisia optimointimenetelmiä. Lopulta hakutaulukoita sovellettiin mallin parametrien hienosäätöön prosessitilojen mukaisesti.

Menetelmää testattiin virittämällä lämmönvaihtimen malli kahteen eri prosessitilaan käyttäen laitokselta kerättyä prosessidataa. Tulokset osoittavat että kehitetty menetelmä pystyi virittämään simulaattorin mallin vastaamaan paremmin dataa useista prosessitiloista. Kuitenkin tarvitaan lisää testausta erityyppisten mallien kanssa, jotta voidaan varmistaa menetelmän yleinen soveltuvuus.

Avainsanat Datalouhinta, parametrien estimointi, klusterianalyysi, dynaaminen simulointi

PREFACE

This master's thesis was done for the NAPCON Strategic Business Unit of Neste Jacobs during the time between the first of January and 31st of July 2017. The work was related to the development of the NAPCON ProsDS dynamic simulator, proprietary of Neste Jacobs.

First of all, I would like to give special thanks to D.Sc. Mikko Vermasvuori and M.Sc. Jyri Lindholm for making this master's thesis possible. I also want to thank my instructors M.Sc. Teemu Liikala and M.Sc. Ville Saarela for their comments and support throughout the thesis process. Especially I would like to thank the examiner of this thesis, Professor Sirkka-Liisa Jämsä-Jounela from Aalto University, for her guidance during the master's thesis project.

I am thankful for my colleagues and friends for their support during this process. Especially I want to thank D.Sc. Stefan Karlsson and M.Sc. Markus Sintonen for their helpful advice.

I gratefully thank my family for their encouragements during this process. Finally, I would like to give thanks to my husband Samuli for his continuous support and love during my studies.

Porvoo, 31st of July 2017

Sanna Hyvönen

TABLE OF CONTENTS

LIST OF ABBREVITATIONS

AHC	Agglomerative hierarchical clustering
BFGS	The Broyden-Fletcher-Goldfard-Shanno method
CLARANS method	Clustering large applications based upon randomized search
CLINK	Complete-linkage method
DBCLASD method	Distribution-based clustering of large spatial databases
DBSCAN method	Density-based spatial clustering for applications with noise
DECODE	Discovering clusters of different densities method
DENCLUE	Density-based clustering method
DFP	The Davidson-Fletcher-Powell method
GG	Gath-Geva fuzzy clustering method
GK	Gustafson-Kessel fuzzy clustering method
LMN	Local model network
LPV	Linear parameter-varying identification
LTI	Linear time-invariant model
MIMO	Multiple-input multiple-output
MINLP	Mixed integer nonlinear programming
MMF	Multimodel framework
ML	Maximum likelihood estimation
NLMS	Normalized least mean squares method
OBF	Orthonormal basis function
OPTICS	Ordering points to identify clustering structure method
PAM	Partition around medoids method
PDF	Probability density function
PWC	Piecewise continuous system
RBF	Radial basis function
RLS	Recursive least squares method
SISO	Single-input single-output
SLINK	Single-linkage method
SNN	Shared nearest neighbor method
SOM	The self-organizing map
SQP	Sequential quadratic programming
T-S	Takagi-Sugeno fuzzy local model
WLS	Weighted least squares method

LIST OF SYMBOLS

A	Heat transfer area
b_{corr}	Correction coefficient
$e(k)$	Model error
g_i	Model coefficients
J_G	Global learning criterion
J_L	Local learning criterion
k	Time step, heat transfer coefficient
M	Number of submodels
$\mathcal{M}_i$	Submodel output
N	Number of observations
$\boldsymbol{P}$	Correction vector
$p(k)$	Scheduling variable
$\boldsymbol{Q}_i$	Weighting matrix
q	Heat transfer
r	Normalized offset
S	Objective function
$\nabla^2 S(\boldsymbol{\theta}^j)$	Hessian matrix
$\nabla S(\boldsymbol{\theta}^j)$	Gradient
ΔT	Temperature difference
$\boldsymbol{u}(k)$	Input
$v(t)$	Disturbance term
w_i	Weight
x_i	State
$\hat{\boldsymbol{x}}(k)$	State estimates
y_M	Model output
$y_p(k)$	Observed process output
$y_u(k)$	Actual process output
θ	Model parameters
θ_0	Process parameters
$\hat{\boldsymbol{\theta}}$	Parameter estimates
$\underline{\Theta_i}$	Parameter vector
λ	Forgetting factor
Φ	Matrix of interpolation factors
$\phi_i(p(k))$	Validity function
$\varphi(k)$	Information space
$\boldsymbol{\psi}$	Data matrix
$\underline{\psi}(k)$	Full data vector

LITERATURE PART

1. Introduction

Process simulators are regularly used in process industry for gaining understanding and evaluating both existing and designed processes. However, model generation tools, such as process simulators, can only generate mechanistic models where the underlying phenomena aexperire known. Often a modeler faces a problem of reliably estimating unknown parameters based on actual process data. [1] Another important aspect is that a typical process plant can operate in several operating states and the model should have enough flexibility to cover the whole operating regime. As Cameron [1] summarized this problem; "*The challenge is not only to develop reliable, efficient and flexible models, but to also develop modelling tools that can help to develop the necessary models in a systematic and efficient manner*."

This thesis aims to answer that challenge by developing a new methodology for tuning simulator models to better match data that contains several operating points. The methodology aims to provide a more efficient way for identifying operating points from data, estimating model parameters and tuning the model to several operating points.

1.1 Background

The use of data mining techniques in manufacturing began in the 1990s and nowadays data mining applications include predictive maintenance, fault detection, quality assurance, scheduling, and decision support systems. The advancement in information technology, data acquisition systems and storage technology, as well as development in machine learning tools, has led to the growing interest of data mining applications for manufacturing processes. [2]

Cluster analysis methods and techniques have been widely applied and adopted for different scientific fields, such as pattern recognition, information retrieval and data mining [3, p. 215]. In the field of chemical engineering, these methods have been adopted for partition of process data to operating states, for example in [4].

Modeling and identification of dynamic systems is an important task and its applications include development of new systems, analysis of existing ones, monitoring, failure prediction, fault detection and design of process controllers. [5] The task of parameter estimation was first solved by Gauss in 1795 when he tried to estimate the orbit parameters of planets [6, p. 204]. Since then the parameter estimation methods have evolved and been adapted to numerous tasks in scientific research.

Previously no known methodology for combining cluster analysis for process state identification with parameter estimation of simulator models has existed. New research is required to study the possibilities of combining cluster analysis of process data with parameter estimation techniques.

1.2 Aim of the work

The aim of this thesis is to develop a new methodology for tuning NAPCON ProsDS models to better represent processes with multiple operating points. NAPCON ProsDS is a dynamic chemical process simulator used in the petrochemical industry. This work is focused on application of cluster analysis techniques together with parameter estimation methods for model tuning with actual process data. Design of experiments, collection and transfer of the process data or development of the mechanistic models are not covered.

The aim of the theoretical part is to review suitable cluster analysis and parameter estimation methods as the basis for methodology development. In addition, review of parameter estimation practices in other process simulators is presented. The main interest of literature study is in acquiring the general picture of suitable methods instead of detailed study of specific methods.

In the experimental part, the methodology is developed based on the literature search. In addition, the methodology is implemented and tested with actual process data on previously build NAPCON ProsDS model. Finally, the results are validated and recommendations on continuation of the work are made.

1.3 Contents

The theoretical part of this work includes chapters 2, 3, 4 and 5. In chapter 2, cluster analysis techniques for operating state identification are presented. Following that, in chapter 3, model parameter estimation techniques are presented with focus on iterative optimization methods. In chapter 4, identification strategies suitable for multiple operating points and recent studies are presented. Finally, in chapter 5, several examples of parameter estimation practices in other simulators are presented and evaluated.

The experimental part of this work includes chapters from 6 to 11. In chapter 6, first the current state of parameter estimation in the NAPCON ProsDS simulator is presented and based on that, the requirements for the new methodology are formulated. The design of the methodology is presented in chapter 7 and the implementation is discussed in chapter 8. The chapter 9 presents the testing and chapter 10 the results. Finally, the conclusions are discussed in chapter 11.

2. Cluster analysis for operating state identification

A typical process plant can operate in several operating states or conditions and different sections and units can be in different states. The state of the unit refers to the values of one or more key variables that can be either input, state, or output variables. Operating state identification can be described as the process of aggregating the state of process unit from its process variables. [7]

Examples of common operating states include plant startup, grade change and shutdown. Different states can be divided into either modes, which refer to constant operation, or transition between modes. Applications of operating state identification include adjusting controller parameters to different states, model identification, fault detection and alarm management. [8]

Cluster analysis or clustering has been applied for operating state identification, for example in [4]. In this chapter, first the general principle of cluster analysis is presented. Then literature survey of clustering algorithms is conducted. Finally, the application of cluster analysis for operating state identification is discussed.

2.1 General principle

Cluster analysis or clustering is an unsupervised classification tool used in several technical and scientific areas including pattern recognition, information retrieval, and data mining. The aim of clustering is to organize set of data points into homogeneous classes. Similarity of objects is often defined by distances, such as the Euclidean distance seen from equation 1. [3, pp. 215-216]

$$\boldsymbol{d}(\boldsymbol{a},\boldsymbol{b}) = ||\boldsymbol{a}-\boldsymbol{b}|| = \sqrt{\sum_{i=1}^{n}(\boldsymbol{a}^i-\boldsymbol{b}^i)^2} \quad (1)$$

Where $\boldsymbol{a}$ and $\boldsymbol{b}$ represent two data points in the Euclidian n-space.

One approach for classifying clustering methods is based on whether and object belongs strictly to only one cluster as in hard clustering or to several clusters as in soft or fuzzy clustering. In fuzzy clustering objects can belong to several clusters with different levels of membership. A typical example is the fuzzy k-means method. [3, pp. 218-220] Another approach is to classify the methods based on the cluster model they apply. Cluster model refers to the notion of cluster and its different properties. [9]

2.2 Clustering methods

As discussed in the previous chapter, clustering methods can be grouped based on the cluster model they apply. In the following subchapters connectivity-based, centroid-based, distribution-based, and density-based clustering methods are presented.

2.2.1 Connectivity-based clustering

Connectivity-based or hierarchical clustering associates each cluster with an index and a hierarchy. Objects in the hierarchical clustering belong to several clusters. A hierarchical clustering algorithm transforms the similarity into a sequence of nested partitions, in such a way that the closest objects are grouped in clusters with smallest indexes. The hierarchical clustering methods can be divided into two approaches, divisive approach and agglomerative approach. [3, p. 224]

The divisive approach starts with one cluster containing all objects. Then, once per iteration, one cluster is split into two clusters. This cluster is selected based on some

similarity rule, such as distance. Iteration continues until every object is alone within a cluster. [3, pp. 224-226]

The agglomerative hierarchical clustering (AHC) approach starts with one cluster per object. Then, once per iteration, closest clusters are merged until only one cluster remains. These clusters are selected based on some agglomerative criterion. An agglomerative criterion describes the dissimilarity measure between groups. Different approaches include the single-linkage criterion, the complete-linkage criterion, and the average-linkage criterion. [3, pp. 224-226]

The tree structure of the hierarchical methods has made them very popular. However, due to the complexity of the AHC algorithm it is not efficient for large sets of data. [3, p. 232] Efficient algorithms for large data sets for the single-linkage method (SLINK) and the complete-linkage method (CLINK) have been proposed in [10] and [11]. The indexed hierarchy tree structure is illustrated in Figure 1.

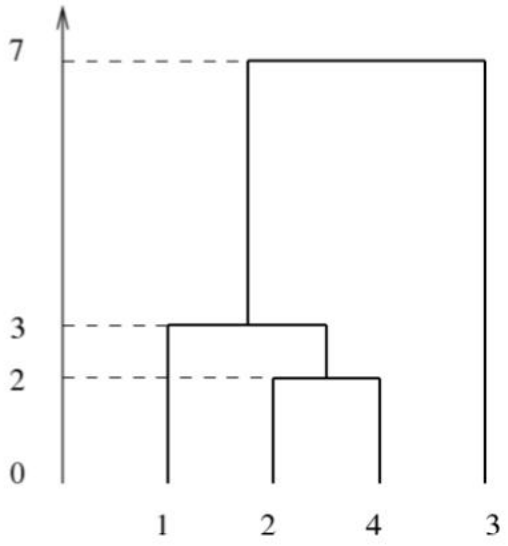


Figure 1. Indexed hierarchy [3].

2.2.2 Centroid-based clustering

Centroid-based methods, such as the family of k-means algorithms represent the clusters as centers, which may or may not be a member of the data set. The k-means algorithm proposed in [12] assigns objects to the nearest cluster center and then updates the cluster centers. A major disadvantage of these methods is the need to know the

number of clusters a priori and the sensitivity to initialization. [3, pp. 233-239] Various alterations have been proposed for the k-means method in the literature.

The k-means++ is a k-means algorithm augmented with a randomized seeding technique proposed in [13]. The algorithm chooses the starting centers of the k-means algorithm at random but weights the data points according to their squared distance from the closest center already chosen. According to the authors, the augmentation improves both speed and accuracy of the k-means algorithm. [13]

The fuzzy k-means algorithm, originally proposed in [14], allows the object to belong to several clusters with varying membership. The algorithm suffers from the same disadvantages as the k-means method; the solution is a local minimum and the results depend on the initialization. [3, p. 241]

Dynamic clustering methods, such as the k-medoids algorithm, proposed in [15], are based on the idea that cluster centers are not necessarily centroids of clusters. Compared to the traditional k-means algorithm, the k-medoids method chooses data points as cluster centers. Different versions of the algorithm have been proposed such as partition around medoids (PAM) [16] and clustering large applications based upon randomized search (CLARANS) [17]. [3, p. 240]

2.2.3 Distribution-based clustering

Distribution-based clustering algorithms, such as the expectation-maximization algorithm, are based on the assumption of statistical distribution model on the data. The algorithm is based on maximizing the observed likelihood function by using the complete likelihood function. An important advantage of the algorithm is that it can handle situations where part of the data is missing. [18, pp. 373-378]

Distribution-based clustering of large spatial databases (DBCLASD) was proposed in [19]. It is based on the assumption that the points inside the cluster are uniformly distributed. It tries to solve issues related to clustering in large databases such as the requirement for minimal number of input parameters, efficiency, and the ability to discover clusters of arbitrary shape. According to the authors, the algorithm determines

the appropriate number and shape of clusters for a database without requiring any input parameters and it is efficient even for large databases. [19]

2.2.4 Density-based clustering

Density-based clustering algorithms define cluster as a set of data objects spread in the data space over a contiguous region of high density of objects. Different clusters are separated by contiguous low-density regions. A data object located in a low-density area is considered to be either noise or an outlier point in the density-based methods. The methods are sensitive to the selection of correct density level. The effect of selected density level is presented in Figure 2. [20]

Density-based spatial clustering for applications with noise (DBSCAN) was first proposed in [21]. Unlike other common clustering algorithms, such as the k-means, the algorithm does not require the number of clusters in the data to be known a priori. Instead, two parameters, the minimum number of points to form a dense region and the maximum radius of the neighborhood from point, are required. The algorithm is able to find arbitrarily shaped clusters, it has notion of noise and it is robust to outliers. [21]

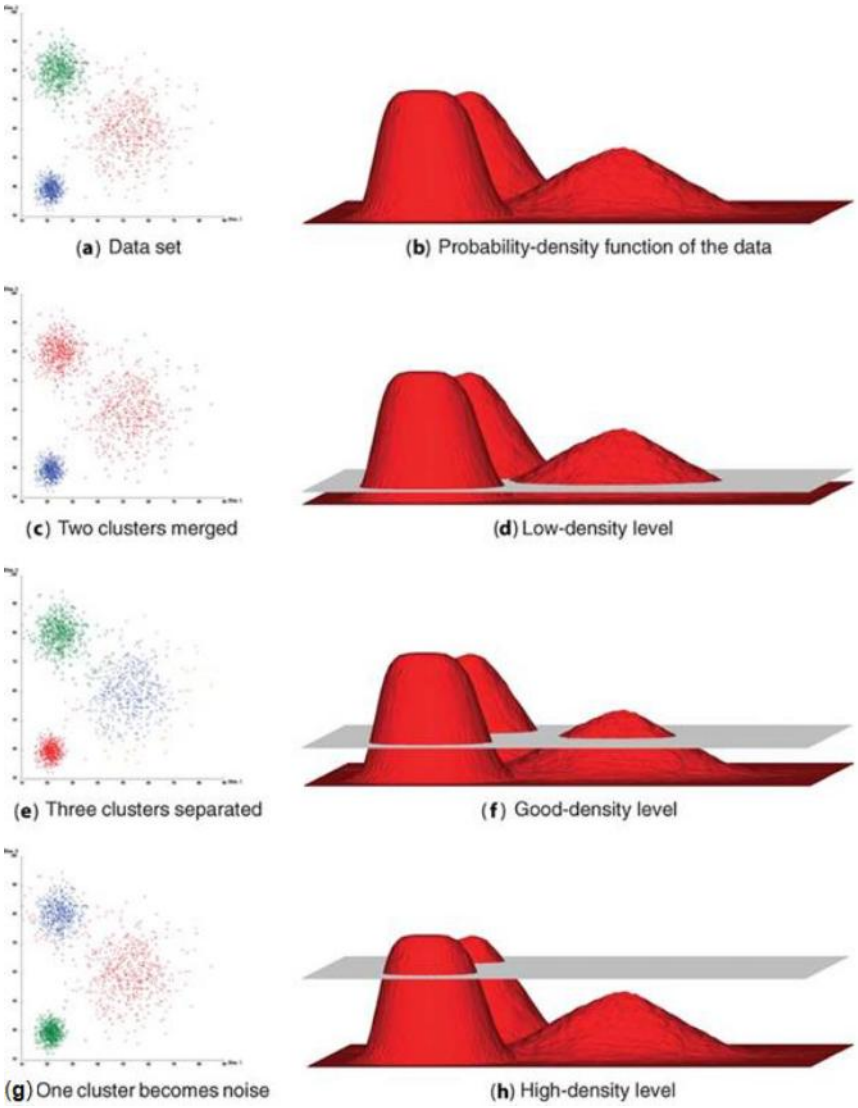


Figure 2. Effect of density level selection in density-based clustering [20].

Density-based clustering (DENCLUE), proposed in [22], uses a kernel density estimator to define clusters. According to the authors, the DENCLUE algorithm can have faster runtime compared to the DBSCAN [22]. Any density estimator can be used such as a Square Wave kernel or a Gaussian kernel. In this method, each data point is associated with a local maximum of the overall density function, called the density attractor and the cluster is defined as a connected component of density attractors. The DBSCAN algorithm can be seen as a special case of the DENCLUE with an uniform spherical kernel. [20]

A major disadvantage of the density-based methods is determining a suitable density level when different regions of the data space have different local densities. For solving this problem hierarchical density clustering methods has been proposed such as ordering points to identify clustering structure (OPTICS), proposed in [23].

The hierarchical density-based clustering algorithms compute clusters at different density levels in a single run. Another alteration, shared nearest neighbor (SNN) [24] algorithm uses a similarity measure based on the number of shared neighbors instead of a traditional distance measure. Discovering clusters of different densities (DECODE), proposed in [25], assumes that the data is generated by different point processes and tries to create clusters as connected regions of points whose distances to their m-th nearest neighbor are similar. [20]

2.3 Process data partition with clustering

Cluster analysis techniques have been utilized for operating region partition as a pretreatment step before model identification. Partition of complete operating space to smaller operating regions is illustrated in Figure 3. [5]

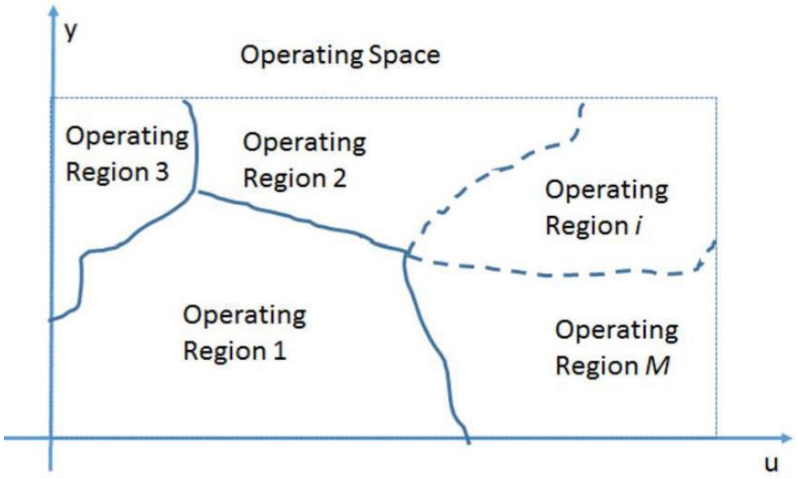


Figure 3. Partition of operating space [5].

In literature, different clustering algorithms have been applied for operating space partition including fuzzy c-means [26], the fuzzy k-means [27], the k-means [28], Gustafson-Kessel fuzzy clustering (GK) [29] and [30], and Gath-Geva fuzzy clustering (GG) [31]. [5] The self-organizing (SOM) or Kohonen map, first proposed in [32], has

also been applied for state identification and process monitoring with industrial applications in [33].

Operating space partition algorithms can be divided into two categories, input space and product space clustering, depending whether only input data or input-output data is used. The input space clustering strategies relay only on the input space data for clustering the process data. This strategy has the disadvantage that the data partition is based only on input data distribution and the process behavior is not considered. Another drawback is the determination of a sufficient number of input variables for clustering the data properly. [5]

The product space clustering strategy jointly considers both input and output data in the clustering procedure. Examples of methods utilizing this strategy are the GK and the GG algorithms. However, these methods are sensitive to initial values and they are unable to identify clusters of unequal volumes. [5]

3. Model identification

Model or system identification can be defined as building mathematical models of dynamic systems based on observed data from the system [34, p. 1]. Goal of system identification is to minimize the deviation between the real process and its derived mathematical model [6, p. 8].

According to [34] system identification requires three basic entities; the data record of the input-output data, the model structure, and an identification method. The data record can be collected from normal operation or from identification experiments. The model structure can be derived from physical laws, in which case the system identification is called parameter estimation problem, and the resulting model is called grey-box model. On the other hand when model structure does not reflect physical consideration, the resulting model is called experimental or black-box model. [34]

Another important aspect of the system identification is the fact that the real process to be identified is always subject to disturbances. Common sources for these disturbances are measurement noise and unwanted inputs. In system identification, it is often assumed that these effects can be lumped into and additive disturbance term $v(t)$. [34]

The general sequence of identification can be formulated as seen from Figure 4. In general, the process data is first preprocessed and then the selected identification method is applied together with the assumed model structure. After the identification, the produced model is validated, and if the result is not acceptable the previous steps are repeated until an acceptable process model is obtained. [6]

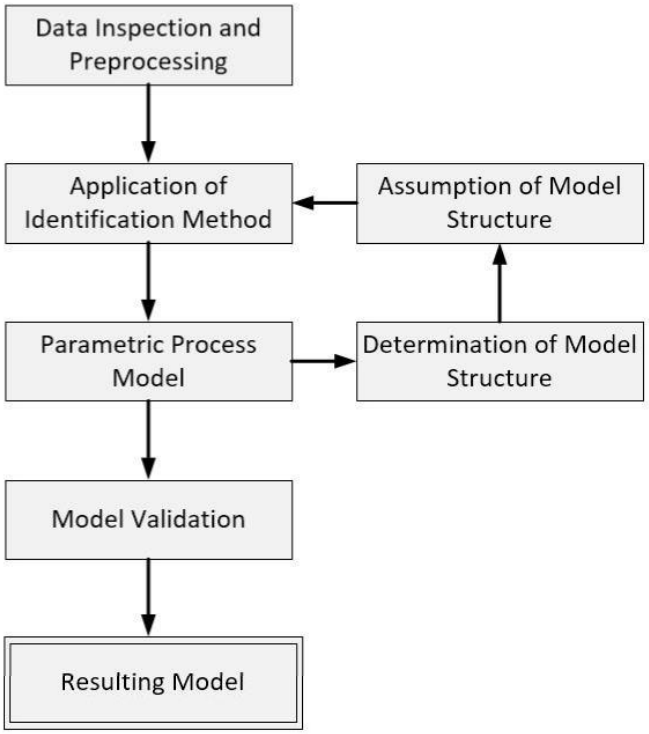


Figure 4. Sequence of identification, adapted from [6].

Model identification methods can be generally divided into non-parametric and parametric methods. Parametric models have a structure and a finite number of parameters, when non-parametric models have neither. Non-parametric models provide a relation between a certain input and the corresponding response. This relation can be presented by means of a table or a characteristic curve. Parametric models are equations which contain model parameters. [6, p. 13]

Non-parametric methods include the frequency response method, the Fourier analysis method, and the correlation analysis method. The frequency response method excites the system with a periodic test signal to determine the frequency response characteristics for linear processes. The Fourier analysis method uses step or impulse

response of the system to determine the frequency response. The correlation analysis method is based on auto- and cross-correlation functions. [6, pp. 15-19]

Parametric models are constructed by assuming a dedicated model structure and using parameter estimation methods to determine the model parameters. Parameter estimation methods apply statistical regression methods to minimize the error between the model output and the data. Iterative-optimization methods can be separated from traditional parameter estimation methods as they apply non-linear optimization techniques to estimate parameters for non-linear systems. Other identification methods utilized for parametric models include subspace-based methods, neural networks and the Kalman filter. [6, pp. 18-19]

Focus on this chapter is on parameter estimation and iterative-optimization methods. In the following subchapters first parameter estimation methods and then iterative-optimization methods are presented.

3.1 Parameter estimation methods

A theoretically derived model might be a very accurate presentation of the process model in form, but without the correct values for the unknown parameters, the model prediction will be inaccurate or even useless [35]. Parameter estimation refers to the process of obtaining values of the model parameters by matching the model output to the measured data. The solution strategy for obtaining the parameters is to minimize a suitable objective function. [35] [36]

According to Isermann, et al. parametric model identification task can be mathematically formulated as follows: for a real process described with parameters θ_0 and output $y_u(k)$, with a known model structure y_M, find the model parameters θ that result in a model which best fits with N observations $y_p(k)$. This can be formulated mathematically as seen from the following equations. [6]

$$\theta_0^T = (\theta_{10}\ \theta_{20}\ \dots \theta_{m0}) \quad (2)$$

$$y_u(k) = f(\theta_0) \quad (3)$$

$$y_M = f(\theta) \quad (4)$$

$$\theta^T = (\theta_1\ \theta_2\ \dots \theta_m) \quad (5)$$

When the known model equations are linear functions of the parameters, the resulting problem is called linear estimation. However, real life processes are often nonlinear functions of parameters and in this case the problem is called nonlinear identification. [35] [36] Two most commonly used parameter estimation methods are the least squares estimation and the maximum likelihood estimation. [37, p. 139]

3.1.1 Least squares

Method of least squares was first solved by Gauss in the year 1795 and he later formally derived the method of least squares in two papers published in 1821 and 1823. The least squares method allows model parameter estimation from possibly noisy signals and it can be used both in static and dynamic systems, non-linear and linear as well as single-input single-output (SISO) or multiple-input multiple-output (MIMO) systems. Modifications of the least squares, such as recursive least squares (RLS) and weighted least squares (WLS) are better fitted for some applications. [6]

In the least squares estimation, the best fit is defined by first defining the model error or residual as $e(k)$ and then determining the minimum sum of squared errors. This problem is formulated as a quadratic cost function *S* seen from equations 6 and 7. [6]

$$e(k) = y_P(k) - y_M(k) \quad (6)$$

$$S = e^2(1) + e^2(2) + \cdots + e^2(N) = \sum_{k=1}^{N}((e(k))^2 \quad (7)$$

The least squares method was first applied for the difference equations of dynamic processes by Kalman in 1958. In the dynamic case the equation error $e(k)$ is interpreted as difference of the new observation $y(k)$ and the one step prediction $\hat{y}(k|k-1)$ of the model. To calculate the parameter estimations, first a data matrix $\boldsymbol{\psi}$ and a model output

vector y are set. Then $\psi^T\psi$ and $\psi^T y$ are calculated. The solution for the time step k can be mathematically formulated as seen from the following equations. [6]

$$e(k) = y(k) - \hat{y}(k|k-1) \tag{8}$$

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{\psi}^T\boldsymbol{\psi})^{-1}\boldsymbol{\psi}^T y \tag{9}$$

$$\boldsymbol{\psi} = \left(\begin{array}{c|c} -y(0) & u(0) \\ -y(1) & u(1) \\ \vdots & \vdots \\ -y(N-1) & u(N-1) \end{array}\right) \tag{10}$$

$$y = \begin{pmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{pmatrix} \tag{11}$$

The recursive least squares (RLS) is often applied for online-identification since it updates the parameter estimates after each new sample and it is computationally light. The method can be formulated as calculation of the new parameter estimate $\widehat{\boldsymbol{\theta}}(k+1)$ as the sum of the old parameter estimate $\widehat{\boldsymbol{\theta}}(k)$ and a correction vector $\boldsymbol{P}(k+1)\boldsymbol{\psi}(k+1)$ multiplied with the equation error. Equation error is interpreted as the difference of the new measurement $y(k+1)$ and the predicted measurement based of the last parameter estimate $\boldsymbol{\psi}^T(k+1)\widehat{\boldsymbol{\theta}}(k)$. This can be formulated as seen from the following equations. [6]

$$\widehat{\boldsymbol{\theta}}(k+1) = \widehat{\boldsymbol{\theta}}(k) + \boldsymbol{P}(k+1)\boldsymbol{\psi}(k+1)e(k+1) \tag{12}$$

$$e(k+1) = y(k+1) - \boldsymbol{\psi}^T(k+1)\widehat{\boldsymbol{\theta}}(k) \tag{13}$$

In the general recursive least squares method, all equation errors have same weight. However, to follow slowly time-varying behavior of the process, the recent measurements must be weighted more strongly than the old measurements. This estimation algorithm is said to have a fading memory. This leads to a method called recursive parameter estimation with exponential forgetting. The algorithm can be used for on-line identification of dynamic systems and the *fading memory* is implemented as a forgetting factor $0 < \lambda < 1$. [6]

3.1.2 Maximum likelihood estimation

Previous parameter estimation methods assumed that the parameters $\boldsymbol{\theta}$ and the observations of the output $\boldsymbol{y}$ are deterministic. An alternative approach is assumption of stochastic signals and this approach is applied in maximum likelihood (ML) estimation. [6] The ML estimator is based on maximizing a likelihood function of the probability density function (PDF) of the data expressed as a function of the parameter to be estimated. [37, p. 139]

The ML method obtains an optimal estimate $\widehat{\boldsymbol{\theta}}$ by maximizing the likelihood function $L(\boldsymbol{\theta}|y)$ of a random variable y. This function is the probability density function of y expressed as a function of the parameters to be estimated $\boldsymbol{\theta}$. Exponential functions of $\boldsymbol{\theta}$ are commonly used as PDFs and therefore a log-likelihood function $l(\boldsymbol{\theta}|y) = lnf_y(y\,;\boldsymbol{\theta})$ is commonly used instead of the likelihood function $L(\boldsymbol{\theta}|y)$. The parameter estimate is obtained by maximizing the log-likelihood function and solution it obtained by setting the partial derivates of the log-likelihood function to zero with respect to $\boldsymbol{\theta}$. This can be formulated as in the following equations. [37, pp. 139-140]

$$\widehat{\boldsymbol{\theta}} = \arg max_{\theta}\{l(\boldsymbol{\theta}|y)\} \tag{14}$$

$$\frac{\partial l(\theta|y)}{\partial \theta} = 0 \tag{15}$$

The ML estimator is widely used as the estimate gives the minimum estimation error covariance and for its efficiency [37, p. 139]. A limitation of the maximum likelihood estimation is high computational burden. However, the maximum likelihood estimators are flexible and can be utilized for many different settings. [6]

3.1.3 Kalman filtering

Kalman filtering is a method that can be used for estimating the states $\hat{\boldsymbol{x}}(k)$ of a discrete-time system based on measurements of the input $\boldsymbol{u}(k)$and the output $\boldsymbol{y}(k)$. A benefit of the extended Kalman filter is that it can be used to estimate both the model states and the parameters. In the Kalman filter, states are first predicted one-step ahead and then the prediction is corrected based on the new measurements of the output $y(k+$

1). The traditional Kalman filter is not applicable to parameter estimation, but it has been utilized for smoothening the measurements used for parameter estimation. [6]

The extended Kalman Filter however allows the parallel estimation of states and parameters for both linear and non-linear systems. The extended Kalman filter linearizes the model around the current estimated operating point. The disadvantage of this approach is that the filter can diverge and go away from the true operating point leading to faulty results. [6]

3.2 Iterative-optimization methods

One approach for solving parameter estimation is using numerical optimization algorithms. These methods allow adjusting the parameters in such way that the model matches best with recorded measurements. Benefit of iterative parameter estimation is that it allows direct determination of physical parameters in non-linear process models. Constraints can be included such as the stability of the resulting system or the requirement that the estimated parameters are positive. [6]

In this chapter, first a general principle of iterative numerical optimization is presented. Then literature survey of the numerical optimization algorithms suitable for parameter estimation is presented.

3.2.1 General principle

The objective function of the iterative parameter estimation is a measure of overall departure of model calculated values from the measurements. For an individual measurement, this can be defined as the residual e that was defined in equation 6. [36, p. 13]

Different objective functions can be selected for the parameter estimation. Often the least squares or the weighted least squares objective is selected, but objective functions applying the generalized least squares or the maximum likelihood estimation can be also used. The weighted least squares objective function can be seen from equation 16. [36, pp. 15, 68]

$$\boldsymbol{S} = \sum_{i=1}^{N} \boldsymbol{e}_i^T \boldsymbol{Q}_i \boldsymbol{e}_i \quad (16)$$

Where $\boldsymbol{Q}_i$ is the user specified weighting matrix.

Numerous general purpose optimization methods can be applied for iterative parameter estimation. These methods can be classified depending on whether they require derivates of the objective function to be evaluated. Direct search methods are derivate free where on the other hand gradient methods require the derivates or approximated derivates. [36, p. 67] In the following subchapters first the derivate methods, divided by whether they evaluate Hessians or gradients, are presented and in the last subchapter the derivate free methods are presented.

3.2.2 Hessian methods

Newton's method is a well-known optimization technique that obtains the parameter search vector with equation 17. The method has a rapid convergence, but it is not guaranteed that it will converge from an arbitrary starting point. In addition, problems arise when the Hessian matrix is indefinite or singular. To overcome issues with the method, several modifications, such as the modified Newton's method, have been presented. [36, pp. 71-76] The method adds a positive diagonal matrix or a full matrix to the true Hessian to ensure that the modified Hessian is positive definitive in all situations [38, p. 141].

$$\Delta\boldsymbol{\theta}^{(j+1)} = -[\nabla^2 S(\boldsymbol{\theta}^j)]^{-1} \nabla S(\boldsymbol{\theta}^j) \quad (17)$$

Where $\nabla^2 S(\boldsymbol{\theta}^j)$ is the Hessian matrix and $\nabla S(\boldsymbol{\theta}^j)$ the gradient of the objective function.

Interior-point methods are constrained numerical optimization techniques that have been applied for linear programming, semidefinite programming and quadratic programming. Interior-point primal-dual methods have been applied for practical applications such as portfolio optimization and optimal control. [39]

3.2.3 Gradient methods

Conjugate gradient methods do not require second derivates and they have relatively small storage requirements and are therefore suited for large scale problems. The

method computes the conjugate vectors using only the previous vector and therefore requires only little storage. Nonlinear conjugate gradient methods include the Fletcher-Reeves method and its modification the Polak-Ribière method. [38, pp. 100-122].

Gradient descent or steepest descent is used to solve non-linear programming problems. The method tries to find the local minimum by taking steps proportional to the gradient. Even though the gradient descent is not the most effective technique when the objective function's closed form is known, it has a low computational burden. The method is sensitive to the selected step size and solution can only be reached if the step size is sufficiently small. [40, pp. 72-76]

Quasi-Newton or secant methods utilize only the first derivatives of the objective function and avoid the calculation of the Hessian matrix. The Davidson-Fletcher-Powell (DFP) and the Broyden-Fletcher-Goldfard-Shanno (BFGS) algorithms are well-known examples of the Quasi-Newton methods. These methods are more rapidly convergent, robust and economical than the conjugate gradient methods, but they require large data storage and are not well suited for large scale problems. The BFGS method is often favored because it is less prone to loss of positive definitiveness or singular problems and it has better theoretical convergence. [36, p. 77]

3.2.4 Derivate free methods

Direct search or derivate free methods are optimization techniques that do not explicitly use derivates. These methods are well suited for problems where explicit information about the gradient of the cost function is not known or it cannot be estimated accurately. A major limitation is that many of these methods are heuristic, and convergence is not guaranteed. In addition these methods have slower convergence rates compared to for example to the Quasi-Newton methods. However they are widely applied for their simplicity and suitability for difficult optimization problems. Common direct search methods include the family of pattern search algorithms and the Nelder-Mead or downhill simplex algorithm. [41]

The pattern search methods include several methods for solving nonlinear unconstrained optimization problems that do not require the gradient of the objective

function to be known or estimated. The method named "pattern search" was originally described in [42] and other variations include coordinate search with fixed step sizes [43], evolutionary operation with factorial designs [44] and more recent multidirectional search method [45].

The procedure of the methods includes conducting a series or a pattern of exploratory moves around the current estimate and then updating the estimate or updating the step size. The usual practice for updating the step length is to continue with steps of same magnitude until objective function does not improve anymore and then half the step size. [46]

The Nelder-Mead algorithm, proposed in [47], is a popular method due to its simplicity and, even though it is considered heuristic. It requires only numeric values of the function and for N decision variables it needs to perform at least (N+1) evaluations of the function per iteration. The method is based on the idea of set of feasible solutions that is referred as the simplex. Once per iteration the poorest solution is replaced with a better solution. [40, p. 83]

The Luus-Jaakola optimization algorithm, proposed in [48], is one of the most reliable direct search methods. It uses random search points and systematic contraction of the search region. An advantage of this method is that it handles nonlinear constraints. A modification that uses multi-pass approach, where the search region is determined by the maximum change of the parameter during last pass, was proposed in [49].

The direct search methods are usually presented only for unconstrained optimization, but some research on applying them for constrained optimization exists. For example inequality bound can be used when for example in the Pattern search the step to out-of-bound are is abandoned is not considered. Another approach is to use an exact penalization approach where the step that is out-of-bound would be penalized with very large value for objective function. [41]

4. State conscious identification strategies

As discussed earlier, continuous chemical process plants operate at different steady states and frequently switch between these states [50]. Typical process automation

applications, such as control, alarm and monitoring systems are normally configured for a single steady-state operation point and do not function properly during transitions. In addition these applications are unaware of their limited applicability domain oblivious to changes in the plant state. [7] This also applies for process models. They are usually simplifications and are designed to only represent some limited operation region properly.

The first step of developing state conscious applications is identification of process states. Once the state identification has been performed, applications or models which can reconfigure themselves to these different states can be constructed. These reconfigurations include turning modules on/off, changing parameter settings and changing models. [7]

This chapter presents literature survey of identification strategies that allow development of operating state conscious process models. Lookup tables can be used to identify models applicable for many operating regions and for nonlinear behaviors and they are discussed in the first subchapter. Then survey of linear parameter varying methods is presented. Finally the more general multimodel identification approach is presented.

4.1 Lookup tables

Model-based representations are concise and can be applied for a wide range of operation, including dynamic behavior, with a small number of model parameters. However, when the functional relationships are not known or are highly complex, lookup tables or maps provide a useful tool for characterizing the system. [51]

Lookup tables can be used to store operating-condition-dependent model parameters [52]. They are especially attractive in applications where computational power and storage capacity are restricted [6]. In this chapter first the general structure of the lookup tables is described, and then identification techniques of these models are listed.

4.1.1 Structure and application

The lookup table models are described as set of data points positioned on a multi-dimensional grid. Height of each scalar point is the estimation of the approximated non-linear function at that point. [6] Lookup tables are generally only used for representing static relationships, although it is possible to represent system dynamics by mapping the rate of change of a given signal of interests onto one of the axes of the table [51].

In real-time applications, the use of lookup tables enables the nonlinear response of the system, within some operating region, to be characterized almost arbitrarily. Lookup tables are extensively used in automotive engine control and monitoring applications. Examples of these applications include Knock spark angle correction and calibration of engine fueling. [51]

Linear lookup tables can be applied for multivariable problems, but each additional independent variable adds a new axis or dimension to the table and the number of parameters increases geometrically. Therefore, these tables are usually only applied for 1 or 2-dimensional problems. The amount of parameters to be identified is however high even for these problems. In addition, in many cases there is need to adapt the identified parameters on-line for slowly time-varying systems. [51] Structure of a 1-dimensional lookup table can be seen from Figure 5.

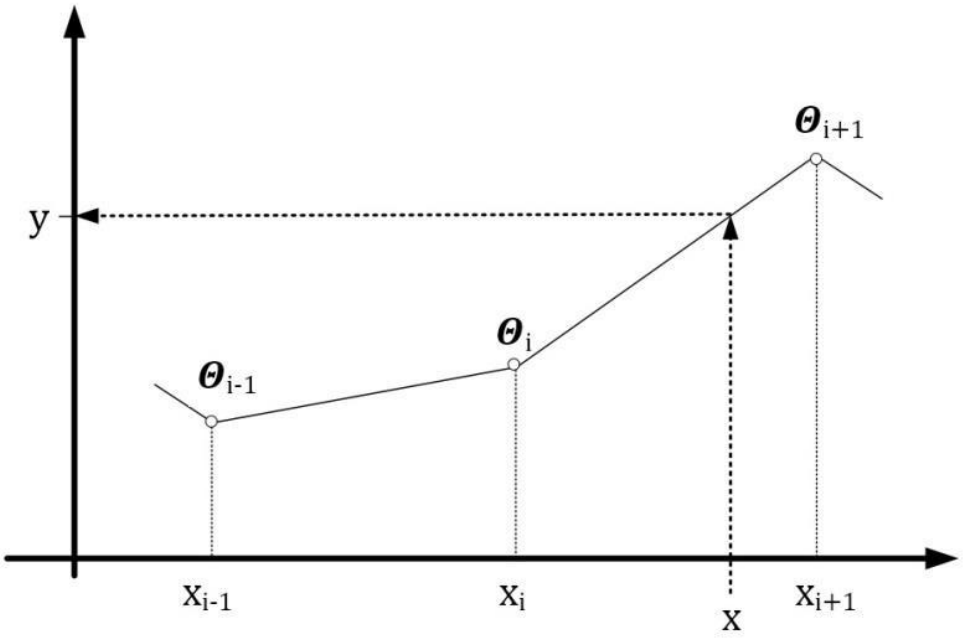


Figure 5. One-dimensional lookup table structure and interpolation, adapted from [51].

As seen from Figure 5, the table is defined by the set of ordered points x_i and the output values y at each of these points are defined by corresponding table parameters θ_i. Intermediate values for the 1-dimensional case are calculated using linear interpolation. A general formulation for calculating the output values y can be seen from the following equations. In the equations $\boldsymbol{y}$ represents the output vector, $\boldsymbol{\Phi}$ is a matrix of input-dependent interpolation factors, r is the normalized offset of x, and $\boldsymbol{\theta}$ is 2-element vector of the table parameters. [51]

$$\boldsymbol{y} = \boldsymbol{\Phi\theta} \tag{18}$$

$$y = \begin{pmatrix} y(1) \\ y(2) \\ \vdots \\ y(n) \end{pmatrix} \quad \boldsymbol{\Phi} = \begin{pmatrix} (1-r(1)) & r(1) \\ (1-r(2)) & r(2) \\ \vdots & \vdots \\ (1-r(n)) & r(n) \end{pmatrix} \quad \boldsymbol{\theta} = \begin{pmatrix} \theta_i \\ \theta_{i+1} \end{pmatrix} \tag{19}$$

In the two-dimensional case, the table output is obtained by interpolating first along the two parallel cell edges in the x_1 direction and then between these two points in the x_2 direction as described in Figure 6. For this case the following equations apply. [51]

$$\boldsymbol{\Phi} = \begin{pmatrix} (1-r_2(1))(1-r_1(1)) & (1-r_2(1))r_1(1) & r_2(1)(1-r_1(1)) & r_2(1)r_1(1) \\ (1-r_2(2))(1-r_1(2)) & (1-r_2(2))r_1(2) & r_2(2)(1-r_1(2)) & r_2(2)r_1(2) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{20}$$

$$\boldsymbol{\theta} = (\Theta_{[i_1,i_2]} \quad \Theta_{[i_1+1,i_2]} \quad \Theta_{[i_1,i_2+1]} \quad \Theta_{[i_1+1,i_2+1]})^T \tag{21}$$

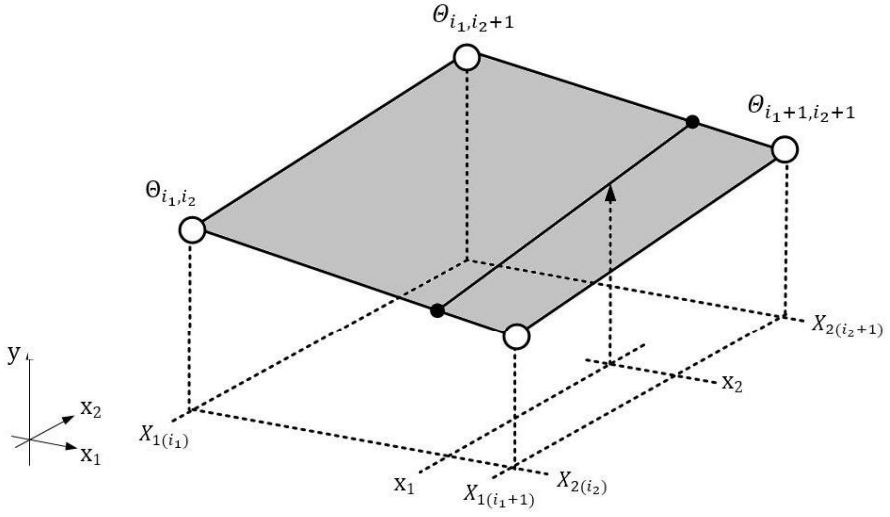


Figure 6.Two-dimensional lookup table structure and interpolation, adapted from [6].

4.1.2 Identification

For the off-line estimation of the lookup table heights, the most widely applied method is to position measurement data points directly on the grid points. With this approach an additional approximation step is not required. However, when the available data points do not correspond to the described positions of the grid, estimation techniques are needed for estimation of the interpolation node heights. [52]

For the off-line estimation of the heights of the interpolation nodes, the least squares method has been proposed in [52]. On the other hand, for the on-line adaptation of the lookup table heights, one commonly used approach is the normalized least mean squares (NLMS) method. However, the authors propose the use of recursive least squares (RLS) algorithm to reduce the convergence time of the estimation. [52]

A weighted RLS estimation algorithm for the look-up table identification has been proposed in [51]. This method is suitable for the 1 and 2-dimensional problems, and the method assumes that the table structure is determined beforehand. According to the

authors, the method should yield unbiased estimates when the regressors are uncorrelated with the noise process and the table structure admits an exact description of the system. However, in real application the linear lookup table is only an approximation of the actual system and therefore the table values are likely to be biased. [51]

The authors [51] further proposed the use of an efficient recursive least squares method that eliminates the need for storing the covariance matrix $\boldsymbol{P}(t)$ that grows rapidly with the dimension of the table. This enables a lighter computational burden of the method. [51] Another approach, a modified RLS algorithm where the covariance matrix $\boldsymbol{P}(t)$ is reinitialized only with some of the parameters, when changing to another interpolation area, was proposed in [52].

4.2 Linear parameter-varying identification

Linear parameter-varying (LPV) identification was proposed in [53]. The LPV identification can be described as a generalization of the classical concept of gain scheduling. The LPV technique is used to describe the nonlinear dynamics of process over a wide range of operating conditions. The parameter variations depend on so-called scheduling signal, which represents the changes in the operating conditions. [54]

The LPV model is linear in the signal relations, but the model parameters are assumed to be functions of time-varying signal; the scheduling variable. The general discrete time dynamic description of the LPV model can be seen from equation 22. In the model equation u represents inputs, y outputs and g_i the coefficients that are functions of the scheduling variable p. [54]

$$y(k) = \sum_{i=0}^{\infty} g_i(p,k)u(k-i) \tag{22}$$

The LPV identification approaches can be divided into two main perspectives; local and global. The local approach identifies multiple linear time-invariant (LTI) models in several operating points and then uses interpolation of the models to cover the entire operating region. On the other hand, the global approach is based on a single data set covering several operating conditions and this data is used to identify functional dependencies of a linear model structure on the scheduling variable. [54]

4.2.1 The local LPV approach

The local LPV approach is based on the interpolation of the local LTI models identified in given operating points. Selection of interpolation method, model structure and operating points are crucial for developing local LPV models. The local LPV approach has been recently applied on distillation columns [55] [56], fermentation processed [57] and on continuous tank reactor [58] [57]. [54]

The choice of operating points defines achievable accuracy of the resulting model but also sets needed number of experiments. Usually adequate gridding is required to represent variation of the local behaviors. Adequate choice of grid points is usually recommended to be 3-6 regarding each dimension of the scheduling variable. [54]

After experiments are conducted at each selected operating point the local LTI models are identified using system identification methods. Often prediction error minimization based system identification methods are used. Then these models are interpolated using for example radial basis functions (RBF), polynomial interpolation or bilinear interpolation. [54]

Four different interpolation schemes are regularly applied in LPV modeling. These schemes include coefficient, output, input and series-expansion based interpolation. Coefficient interpolation scheme interpolates the local models based on their model coefficients. Similarly output interpolation utilizes the weighted outputs of the local model. The last scheme relies on interpolation of the local models in series expansion form instead of input-output form. [54]

4.2.2 The global LPV approach

The global LPV approach identifies the model in one step based only on a single data set with varying scheduling trajectory. Almost all global methods require a linear parametrization of each model coefficient function in terms of a priori chosen set of basis functions. However, the selection of these basis functions often requires a complicated analysis of a first-principles model. This can be solved by non-parametric methods [55] [59]. [54]

Examples of classical parametrization approaches are prediction error minimization, and orthonormal basis functions (OBF) based global estimator. On the other hand, an example of a non-parametric method is the LPV least-square support vector machine (LPV LS-SVM) [55] that can approximate nonlinear functional dependencies directly from data. [54]

4.3 Multimodel identification

The multimodel framework (MMF) has been studied in [60] [61] [62] for modeling and identification of complex, nonlinear and uncertain systems. In this approach the global system model is formed by a set of local models integrated with different degrees of validity. Each model represents system dynamics around specific operating space or operating point [63]. The framework is recognized for its simplicity and transparency, and it allows use of well-known modeling analysis and control design techniques. However the MMF approach has been criticized for creating suboptimal and input depended models [64]. [5] Figure 7 describes the general MMF network structure.

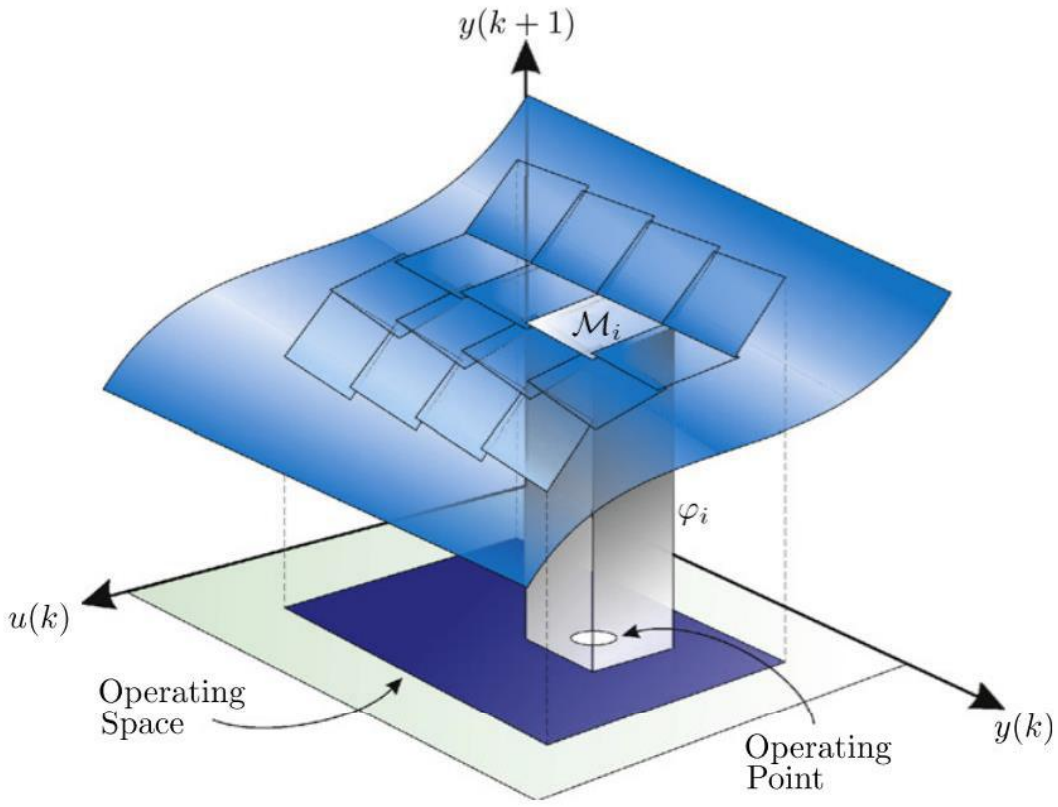


Figure 7. Multimodel network structure [65].

Different fields where the MMF approach has been applied include process optimization, prediction, fault detection, state estimation and controller design. Application examples include control of tubular heat-exchanger [66], identification of the dynamics of a nonlinear centrifugal compressor [67], fault detection of a centrifugal pump [68], control of pH neutralization plant [69] [70] [71] [72], and identification of biochemical reactor [73].

The general multimodel formulation can be presented as in equation 23. The model output $y(k)$ is a sum of the M submodels $\mathcal{M}_i$ output weighted with the corresponding validity function $\phi_i(p(k))$. The variable $p(k)$ is the scheduling variable and it is subset of the information space $\varphi(k)$ that defines the operating space of the system. [5]

$$y(k) = \sum_{i=1}^{M} \mathcal{M}_i(k)\phi_i(p(k)) \tag{23}$$

Various different methods fall under the multimodel framework. These methods include the regime-based multimodel [74], local model networks (LMNs) [75], local radial basis function networks [76], a Takagi-Sugeno (T-S) fuzzy local model [77], and a piecewise continuous (PWC) system [78]. The different methods can be categorized depending on four features: method of realization, partition strategy, submodel identification method and selection of validity function. [5] In the following subchapters these features are discussed.

4.3.1 Global network structure

Method of realization defines how the submodels are combined or interpolated along with their validities to form the global system output. A common implementation is to weight the submodel outputs, which is presented in Figure 8. Another possible implementation is to use weighted parameters, which is presented in Figure 9. [65]

In the first approach the output of the network is a weighted sum of local model outputs, where weights w_i can be scheduled using some scheduling vector. This method can be formulated as in equation 24. This approach allows the multimodel network to contain different local model structures. However, the drawback is that this reduces the transparency of this realization. In addition all the submodels must be stable, or the global network can become unstable. [65]

$$\hat{y}(k+1) = \sum_{i=1}^{M} w_i \, \hat{y}_i(k+1) \tag{24}$$

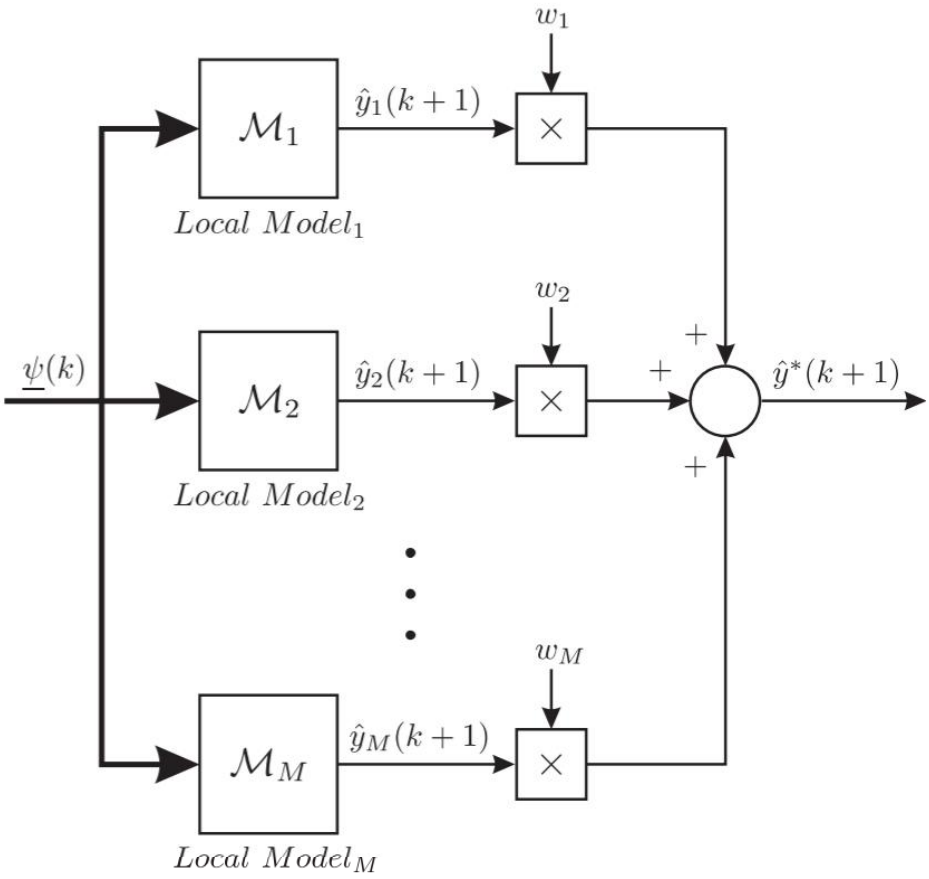


Figure 8. Weighted output realization [65].

Another approach is to have same structure for all of the submodels and blend parameters of the local models to get the global model. Therefore the global model will have the same structure as the local models and it is a valid physical representation of the underlying system, increasing the transparency of the model. This approach can be formulated as in equation 25. [65]

$$\hat{y}(k+1) = \underline{\psi}(k)^T \sum_{i=1}^{M} w_i \, \underline{\Theta}_i \tag{25}$$

Where $\underline{\Theta}_i$ is the parameter vector and $\underline{\psi}(k)$ is the full data vector.

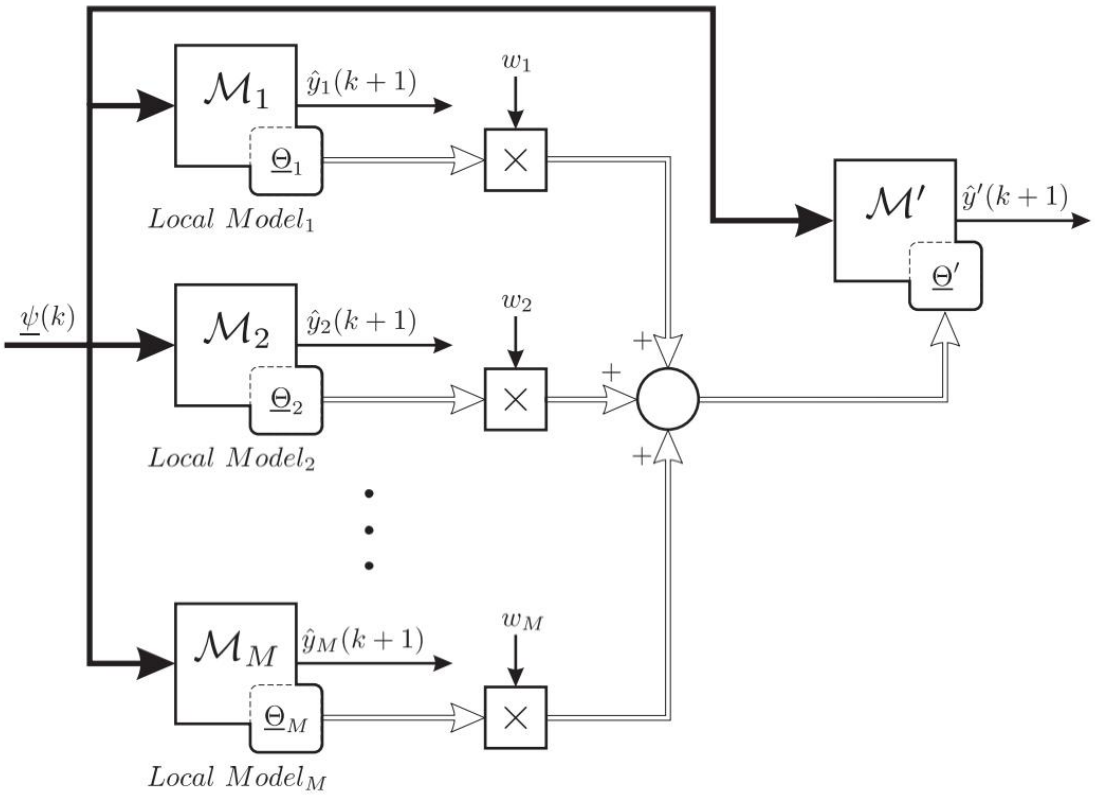


Figure 9. Weighted model parameters realization [65].

4.3.2 Partition of the operating space

One of the main differences between various approaches is the operating space partition strategy. The partition strategy defines the operating region and the structures of the locally formed models. Popular strategies include operating point, axis-orthogonal, axis-oblique and clustering partition. Partition strategies can be categorized by whether they utilize prior knowledge of the system. [5]

The prior knowledge-based partition strategies include model-based and experimental-based approaches. The model-based partition takes into account the prior knowledge of the system's model where the experimental-based partition assumes prior knowledge of the operating conditions and executes experiments in the known operating regions. [5]

The nonprior knowledge-based or data-based strategies include incremental partition algorithms and clustering algorithms. The incremental algorithms partition the system by iteratively adding a submodel once per iteration based on some rules. [5] The clustering partition has been presented in more detail in chapter 2.

4.3.3 Identification of submodels

Submodel structural identification involves the process of associating the subspaces and the local models. The submodels can be either homogeneous or heterogeneous, since the submodels can have different structures. The homogeneous submodels have the advantage, that they can use same learning and optimization techniques [79]. On the other hand, the heterogeneous submodels are more flexible and can cope with the curse of dimensionality [79]. Examples include linear, nonlinear, mechanistic, empirical, neural networks, polynomial, hybrid models and even Gaussian process models. [80]

Parameter estimation of the local models can be either done with a global or a local learning cost function [65]. In global learning the objective is to minimize the error between the system's output and that of the multimodel's output. General expression of the global learning criterion can be seen from equation 26. The global learning criterion is accurate for well-chosen submodel structures, but obtaining appropriate structures can often be a challenge. In addition this approach is computationally heavy and the produced submodels are harder to interpret. [81]

$$J_G = \frac{1}{2}\sum_{k=1}^{N}(\hat{y}(k) - y(k))^2 \quad (26)$$

Where N is the number of data points, $y(k)$ is the actual system output and $\hat{y}(k)$ is the model output.

The local learning approach focuses on locally useful information extracted from the data. It tries to minimize the error between the system's output and all the local model's outputs and produces independent estimation of the parameters of each submodel. This method can be seen from equation 27. Local learning performance is better than the global criterion, but it has the disadvantage of discarding the useful global information from data. [80]

$$J_L = \frac{1}{2}\sum_{i=1}^{M}\sum_{k=1}^{N}\phi_i(k)(\hat{y}_i(k) - y_i(k))^2 \quad (27)$$

Where M is the number of submodels, $\hat{y}_i(k)$ is the submodel output and $y_i(k)$ is the actual system output.

To overcome the disadvantages of both of these approaches, a combined criterion has been proposed. The combined criterion can be seen from equation 28. [80]

$$J = \alpha J_G(\theta) + (1 - \alpha) J_L(\theta), \quad \alpha \in [0, 1] \tag{28}$$

For solving the optimization criteria, different algorithms can be used and the selection depends on the submodel structure. Commonly used identification algorithms include the least squares and the recursive least squares (RLS). [80]

4.3.4 Validity function determination

The validity function defines the transition method between models. The validity function determines how each submodel contributes to the total model and it is therefore crucial for the accuracy of the model. The transition can be either hard or soft. For hard switching it is required that $\phi_i(z(k))$ is either 0 or 1. On the contrary, in soft switching the validity function can also produce numbers between 0 and 1. [5]

Two approaches for the validity determination can be found from literature; prevalidity computation and post-validity computation. Both approaches determine the scheduling variable vector, which again defines the operating region. [80]

In prevalidity computation, determination of the validity is done during the partition of the operating space and it depends on the selected partition strategy. It can be employed directly in the estimation of the local model parameters, for example by using the weighted least squares estimation, where the weights represent the validities. Other example of prevalidity computation is the Gaussian function. [80]

On the other hand, in post-validity computation, the validity is determined after the submodel identification and does not therefore depend on the partition strategy. Examples of this method include polynomial function, simple and reinforced residues, and Bayesian. [80]

5. Parameter estimation in simulators

In former chapters the theory behind various identification methods has been presented. As a continuation, a literature study of parameter estimation applications for simulators is presented in this chapter.

5.1 JModelica.org

JModelica.org is a Modelica-based platform for optimization, simulation and analysis of complex dynamic systems. JModelica.org was developed as the result of research at the Department of Automatic Control in Lund University. [82]

JModelica.org allows parameter estimation for dynamic systems and it can be also used for comparing model outputs to measurements outputs. This is implemented by setting the start values of the model states to correspond with the states at the start of the measurement data. Then a matrix containing the input trajectories is created from the measurement data and it is used as the input to the model during simulation. Then the results of the simulation are plotted against the data. If there is mismatch, parameters can be estimated from the data. [82]

First, the user selects the parameters to be tuned. Usually parameters which are not known exactly and affect the mismatch between model and data are selected. The cost function of the parameter estimation is the squared sum of the difference between the measured profiles and the corresponding model profiles. The parameters are given initial guesses as well as bounds. Then the parameter estimation problem is solved and the model is again plotted against measurement data. [82]

Another interesting application is a grey-box modeling toolbox for JModelica.org. This toolbox is used to develop grey-box models combining a white-box model structure with parameter estimation. The toolbox is composed of four different modules: a Modelica library of models, various files specifying the model components and which parameters to estimate, JModelica.org as the layer for compilation, formulation and solution and a Python module which contains the user interface. [83] This structure can be seen from Figure 10.

The parameter estimation problem is formulated as a dynamic optimization problem and the objective function of the optimization is to minimize the integrated quadratic deviation of the model output from the corresponding measurement data. Gradient-based method is used for solving the optimization problem. [83] Workflow of the toolbox can be seen from Figure 11.

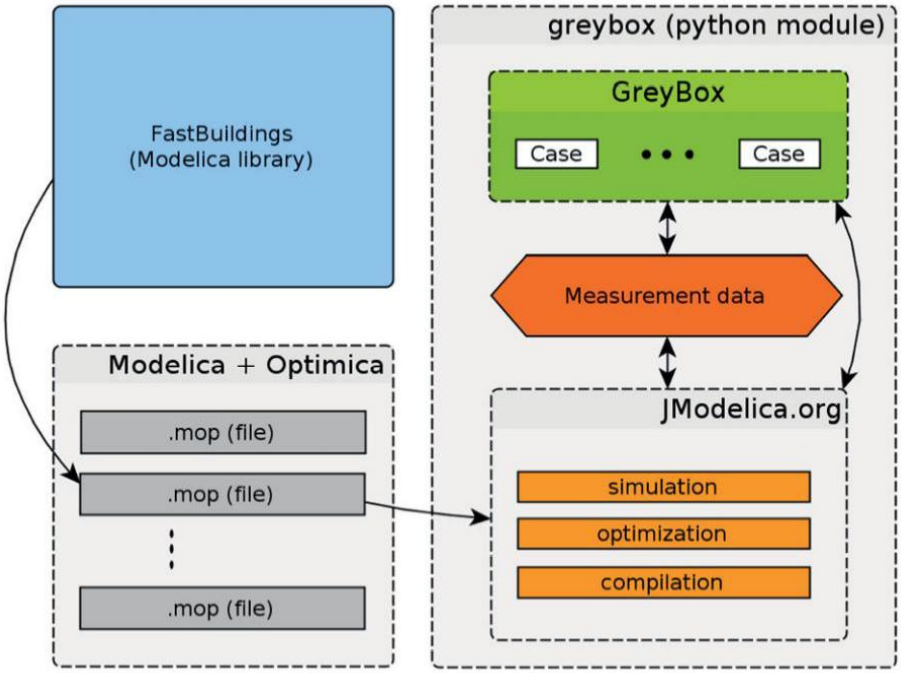


Figure 10. Toolbox for parameter estimation for building models [83].

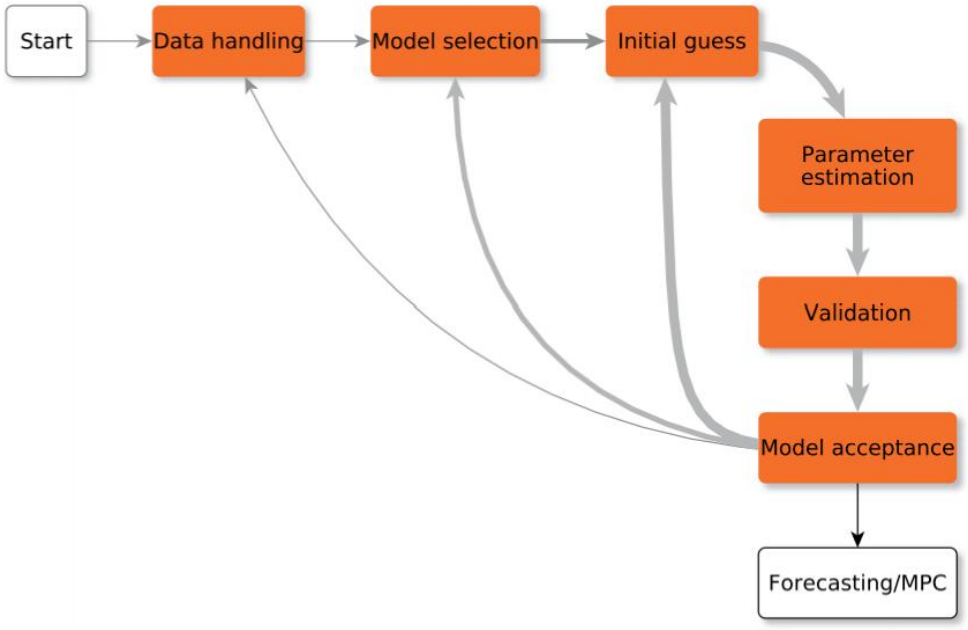


Figure 11.Toolbox workflow [83].

5.2 The Dakota toolkit

The Dakota toolkit is an extensible interface between simulation codes and iterative analysis methods. It contains algorithms for parameter estimation and it uses the nonlinear least squares method. Dakota utilized the Gauss-Newton algorithm and its modifications for solving the nonlinear least squares optimization problem. [84]

The analysis script performs several tasks when estimating the parameters. First the measurement values are used for computer simulation, and then the results of the computer simulation are used for computing the difference between each computed simulation value and the corresponding measured value. Finally these residuals are used for estimating the parameters with the Gauss-Newton algorithm. However, the Gauss-Newton algorithm requires that the user also specifies the gradients of the function with respect to the parameters being calibrated. [84]

5.3 ChemCAD

A methodology for estimating pervaporation unit parameters in professional flowsheeting simulator (ChemCAD) has been proposed in [85]. The proposed

methodology involves the following steps: first measurements were collected from laboratory experiments, then estimation of the parameters was handled as a mixed integer nonlinear programming (MINLP) problem, and finally the simulator was applied for double verification of the parameters. [85]

The parameter estimation was done by writing the mathematical pervaporation model into a program in general mathematical optimization program environment. Then the parameter estimation problem was constructed as a MINLP problem and solved. The ChemCAD simulator was then used for verification. The estimated parameters were implemented into the model and then the simulator was used to recalculate the data measured on the laboratory equipment. Then the model data was compared to the measured data to verify that the parameters were estimated correctly. [85]

5.4 ProSimPlus

ProSimPlus is a steady state process simulation software. The ProSimPlus simulator has an implemented optimization module that uses sequential quadratic programming (SQP) algorithm. The SQP algorithm is an iterative method for nonlinear optimization. [86]

The optimization algorithm has been implemented in the simulator as an optimization module called OPTI. This module minimizes the criterion function by adjusting selected process parameters. The criterion function is also implemented as a module that is connected to the optimization module with an information stream. In addition the optimization module has to be connected to the process blocks to adjust their parameters. The configuration of OPTI module has been presented in Figures 12 and 13. [87]

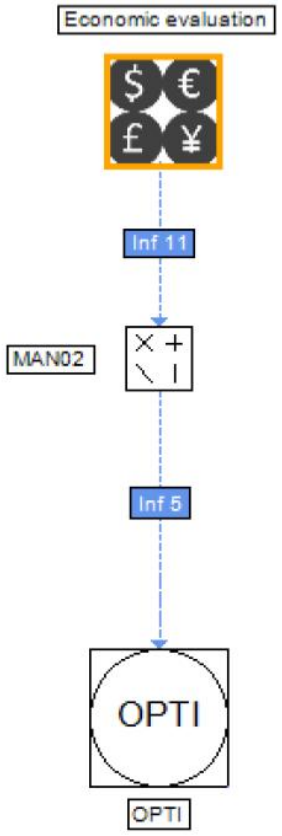


Figure 12. Connection of the optimization criterion to the OPTI module [87].

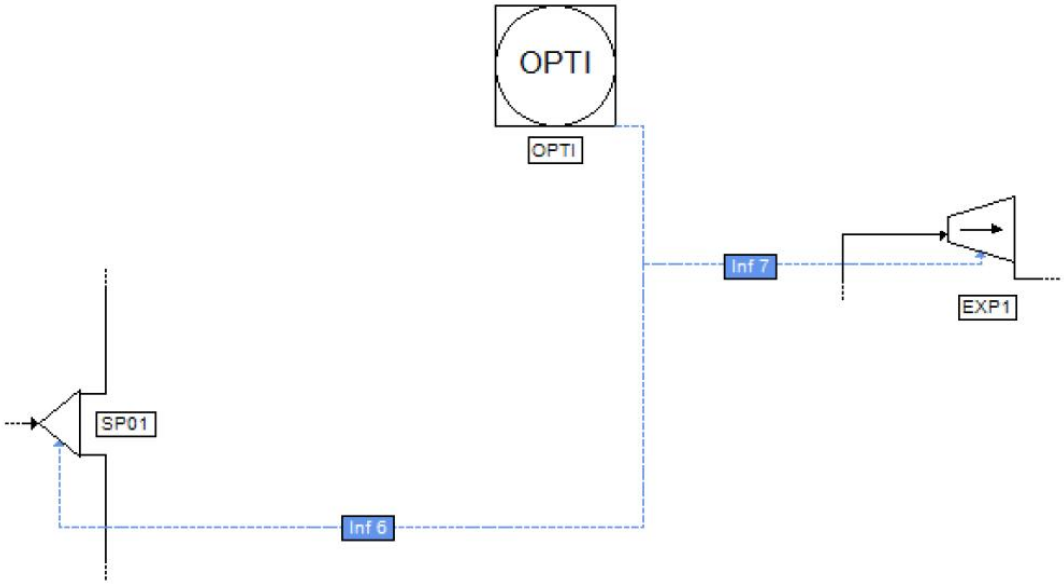


Figure 13. Connection of the OPTI module to the adjusted process blocs and associated parameters [87].

EXPERIMENTAL PART

6. Aim of the experimental part

The aim of this thesis is to develop a new methodology for tuning NAPCON ProsDS simulator models to several operating points based on process data. As the basis for this, the current state of the parametrization process is presented in the first subchapter and then the requirements for the new methodology are discussed.

6.1 Current state of parameter estimation

NAPCON ProsDS is a dynamic process simulator that is used in the petrochemical industry. It has been utilized for dynamic simulation studies, safety calculations and operator training simulators. A typical NAPCON ProsDS model holds thousands of process blocks and therefore determination of the model parameters is a crucial and time consuming step in the model development. However, before this thesis the simulator did not have any specific tool for this task and all the parametrization tasks were conducted manually in an empirical manner.

The data preprocessing included collecting data from the real process, deciding on the operating regime for the model and identifying reference states from the data. Typically, the reference points were manually selected from the data, but in addition some machine learning techniques had been tested. These techniques included clustering algorithms such as the k-means++, the kernel k-means and the DBSCAN algorithm. In addition data analysis algorithms such as principal component analysis, histograms and scatterplots had been tested for visualizing the data.

After the data partition the modeler decided the model parameters to be adjusted, often by trial and error manner. For steady state models, first the mean of variable values were calculated from the selected reference points. Then the model was set up to state corresponding to the current reference point. The model parameters were adjusted in an iterative manner by comparing the model output to data visually and numerically. In addition the stability of the model was monitored. This procedure was repeated for every reference point.

After the parameters were identified in all of the reference states, the model was either set to one reference state or a parameter transition schedule was programmed manually. This parameter transition schedule changed the parameters when the state of the model moved from one reference point to another. After the parametrization the model was tested in the operating regime as well as in exceptional cases outside of the operating regime. If the results of the test were not acceptable, the parametrization process would be done again.

For dynamic models the difference of the model dynamic response and reference data was integrated and the result of integration was used as the measure of good fit. This comparison could also be done visually. Then the parameters were adjusted in a similar manner as in the steady state situation. After the parametrization, a document containing the basis of tuning, the reached parameters, the used methods and the field of applicability was written and attached to the model for future reference. The complete model tuning work process is illustrated in Figure 14.

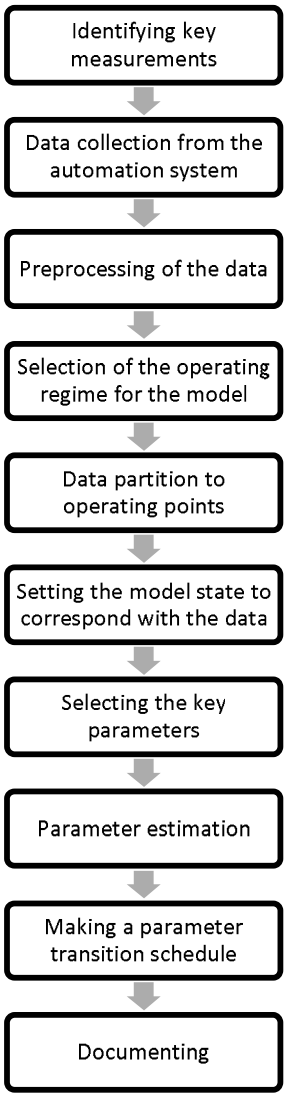


Figure 14. Model tuning work process.

6.2 Design objectives

The main objective for the methodology design was to reduce need for manual work by selecting suitable methods and algorithms for partly automating the model tuning process. Based on the current state presented in the previous chapter, it was decided that the designed methodology would offer methods for process data partition, parameter estimation and parameter transition schedule generation. Other tasks in the work process were dependent on human judgement or were too case specific. The methodology would focus on steady state operating modes, because often only steady state data is available from industrial processes.

Several objectives for the methodology development were identified. The data partition techniques should be well documented and easy to use for modelers not familiar with the algorithms. Secondly to improve the modelers understanding of the process and make the partition process more transparent, data visualization techniques should be used in the partition.

For automating the model tuning in the NAPCON ProsDS environment a parameter estimation tool was required. The tool should, after user configuration, handle the parameter estimation automatically. In addition it should enable generation of a parameter transition schedule based on the estimated parameters. Finally the tool should present the parameter estimation results visually and numerically to the user for documentation and evaluation of results. The previously identified objectives for the methodology design were listed to Table 1.

Table 1. Design objectives for the methodology.

Objectives	Deliverables
Improved understanding of the data.	Offer techniques for data analysis and visualization.
Unsupervised partition of the data.	Offer techniques for unsupervised data partition.
Automated parameter estimation.	Offer tool for automatic parameter estimation in NAPCON ProsDS environment.
Automated parameter transition schedule generation.	Offer tool for automatically generating transition schedule based on the estimated parameters.
Present parameter estimation results visually and numerically for evaluation of results.	Offer tool for visual and numerical inspection of parameter estimation results.

7. Design

Based on the design objectives identified in the previous chapter, suitable methods for the methodology implementation were selected. The selection of suitable techniques is discussed in more detail in the following subchapters.

7.1 Data partition to operating points

For the data partition task two objectives were previously identified; visualization of the data and unsupervised data partition to operating regions. In data analysis graphical views such as scatterplots and histograms are often used for analyzation of the data as well as for the visualization of the possible clusters [3, pp. 218-219]. These techniques were selected for the methodology implementation to make the clustering process transparent and the results easier to evaluate. These methods can also be used for analyzing the data before the data partition to improve understanding of the process conditions.

As discussed in chapter 2, various clustering algorithms have been applied for partitioning the process data to operating regions or points. For the methodology implementation hard clustering techniques are more appealing because they are transparent and easier to interpret. For the same reasons partition clustering techniques are more attractive than hierarchical clustering algorithms. Because process data is often stored in large databases, the selected algorithms should be suitable for large data collections.

For the implementation of the designed methodology the DBSCAN algorithm, the k-means++ and the kernel k-means were selected as the clustering methods, because these algorithms have been previously implemented in the open source numerical computation environment Scilab. Of these methods the DBSCAN seemed most promising based on the literature survey. As discussed earlier, the method is able to detect arbitrary shaped clusters, does not require a number of clusters beforehand and it is suitable for large databases. In addition the method should be able to discriminate outliers or noise from the data. Possible disadvantages include the difficult determination of a suitable density-level.

7.2 Parameter estimation procedure

The objectives for the parameter estimation task involved automatic parameter estimation, generation of a parameter transition schedule and visualization of results. First the parameter estimation methods were selected for the methodology

implementation. Then the method for parameter transition schedule implementation was selected.

7.2.1 Parameter estimation

In chapter 3 several parameter estimation methods were presented. In addition several examples of parameter estimation tools in simulators were presented in chapter 5. All studied simulators utilized iterative-optimization methods for parameter estimation. This is reasonable because often the numerical form of the objective function is not available and these methods are suitable for automatic parameter estimation. Numerical optimization is best suited technique for the task since it does not depend on the model and it can be utilized for steady state, dynamic, linear and nonlinear situations. Therefore, it was selected that an iterative parameter estimation method would be used.

As discussed in the literature survey, several general purpose numerical optimization algorithms can be applied for the iterative parameter estimation problem. Of these methods, the gradient methods and the derivate free methods are most popular. The gradient methods are proven to converge and have faster convergence than the derivative free methods. On the other hand the derivate free methods are easy to implement, require only the function values and are applicable for oscillating models. One challenge with the gradient methods is the calculation of the gradient when the numerical form of the objective function is not available.

In literature [41], several methods have been discussed for calculating the gradient in simulator environments. Manually calculating the gradient from a known numerical form of the objective function is a robust method. However, this approach requires that the user has access to the source code and is able to provide the gradient. [41] This approach is not feasible in this case, since for most NAPCON ProsDS simulator users the source code is not available for inspection.

In some cases, automatic differentiation tools can be applied for obtaining the derivates. Automatic differentiation refers to techniques that use the computational representation of a function to produce analytic values for the derivates [38]. However, this approach is limited when simulation codes mix different programming languages [41] The

NAPCON ProsDS dynamic simulator utilizes two rare coding languages, Common Lisp and Fortran and no known automatic differentiation methods are available for these languages.

Third approach is to estimate the derivates with finite-differences method to obtain the gradient estimates [38]. However, this approach can only be applied for smooth objective functions and precise calculated values [41]. In addition, this approach is limited due to noise, non-smoothness and oscillations in the simulation that cause the estimates with a small finite-differences interval to be highly inaccurate [41]. The NAPCON ProsDS models are often complex entities and oscillations are normal. In addition the simulation speed of complex models is limited and the gradient estimation adds additional simulation tests increasing the optimization time.

According to the author [41], for slow and possibly oscillating simulations the direct search optimization methods are recommended for optimization in simulators. As discussed in chapter 3.2.4, direct search methods, even though often heuristic, are applicable for optimization in difficult situations, where gradient methods are impractical.

Based on the discussion it was decided that using the gradient methods is impractical for the parameter estimation task in the NAPCON ProsDS simulator. Therefore the direct search methods were selected for iterative parameter estimation.

The pattern search algorithm presented in the literature part is attractive for its simplicity and even though the method is quite slow, it is robust. For several decision variables the Nelder-Mead algorithm is attractive alternative because it has a low number of needed function evaluations. Both of these algorithms were selected to be implemented as a part of the parameter estimation tool.

7.2.2 Parameter scheduling

As discussed in the discussion of the current state, to cover the whole operating regime, the model parameters need to be adjusted to the different operating states. The objective of the parameter scheduling is to improve performance of simple models by adjusting

key parameters based on some scheduling variable that describes the current state of the system.

In the literature study, several approaches for identifying process state conscious models were presented. Since, in the simulator context, the model structure is already determined and only the model parameters need to be identified, the lookup table approach was selected for implementation. As discussed, lookup tables are easy to implement, transparent and can be used to store process condition-dependent model parameters.

8. Implementation

The designed methodology was implemented by developing a new procedure for data partition and developing a new parameter estimation tool for the NAPCON ProsDS dynamic simulator. For implementation of the methodology, the current software environment needs to be considered and therefore it is presented in the first subchapter. Then the implementation of the data partition procedure is presented. Finally, the implementation of the new parameter estimation tool for the NAPCON ProsDS dynamic simulator is discussed.

8.1 Software environment

The new methodology is tested in a software environment consisting of several separate programs. The process data collection is handled by separate software for data collection, the data analysis is tested in Scilab environment and the parameter estimation is implemented as a new feature in the NAPCON ProsDS simulator. Because of this the data needs to be transported manually between the environments. This testing state is presented in Figure 15.

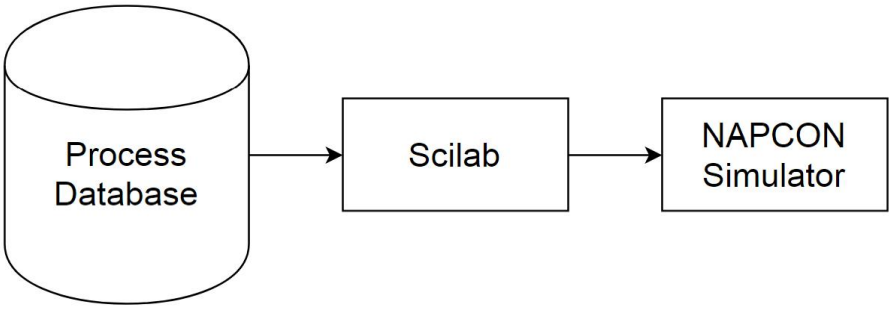


Figure 15. Testing enviroment.

8.2 Data partition procedure

As discussed in the chapter 7, the selected visualization and clustering algorithms have been previously programmed in the Scilab software environment. The developed methodology was tested with the setting seen from Figure 15, but the aim is to later transfer the tested methods to the NAPCON software environment. The testing procedure of the data partition is presented in Figure 16.

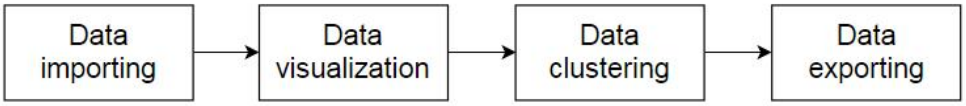


Figure 16. The data partition testing procedure.

8.3 Parameter estimation tool

The parameter estimation tool was implemented as new feature in the NAPCON ProsDS dynamic simulator. The selected direct search optimization methods were implemented as new functions in the simulator. Then the parameter estimation functionality was implemented as a part of the new parameter estimation tool. The parameter estimation functionality is illustrated in Figure 17.

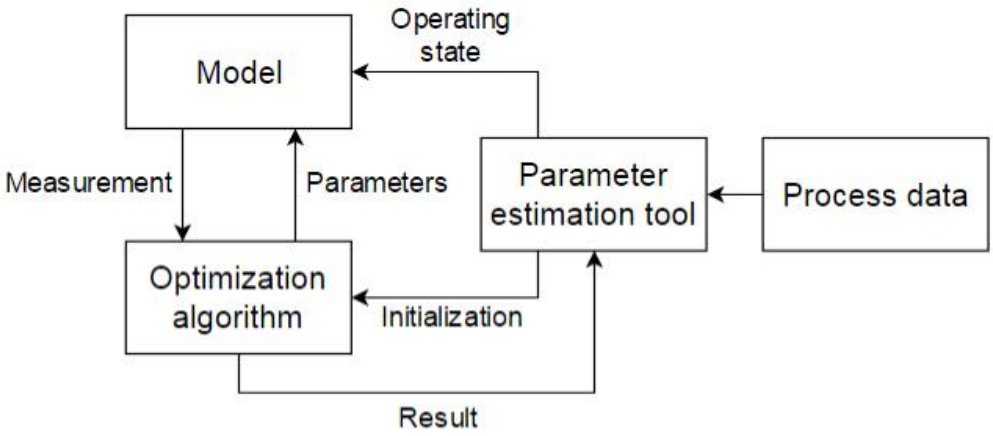


Figure 17. Parameter estimation functionality.

As seen from the previous Figure, the optimization algorithm was separated from the parameter estimation tool. This enables flexibility in the selection of the optimization algorithm and in addition new algorithms are easy to implement. The complete parameter estimation procedure is illustrated in Figure 18.

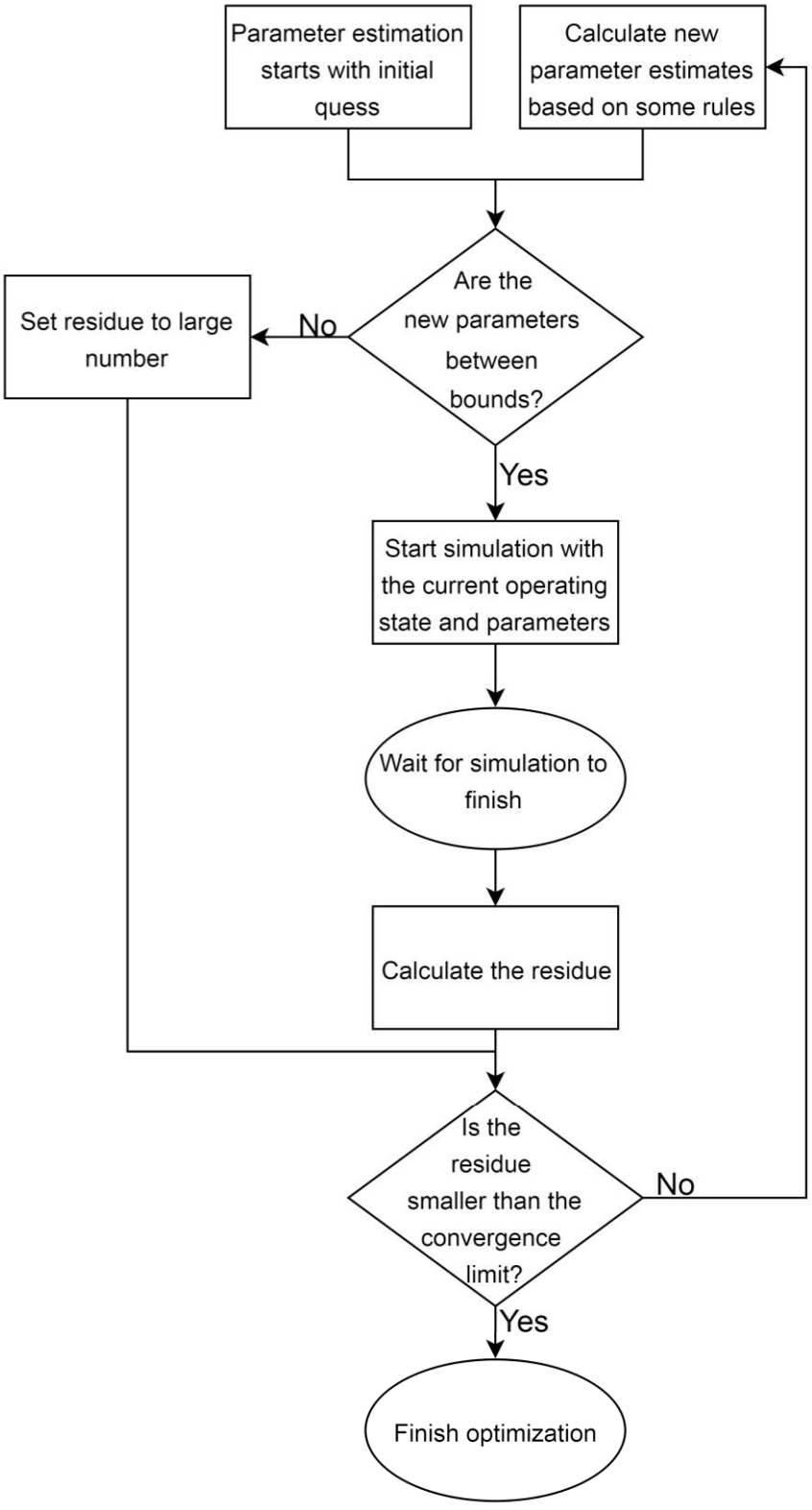


Figure 18. The parameter estimation procedure.

As seen from the previous figure, the parameter estimation tool allows implementation of parameter bounds. These bounds are implemented with a penalty function. In addition to the convergence limit, the procedure allows several stopping rules such as the maximum number of iterations for the optimization.

Based on the literature study of existing parameter estimation tools, it was selected that the tool would be implemented as a model block that contains the parameter estimation functionality. The parameter tuning block enables estimation of several parameters and controlling several input state variables. For easy configuration each signal has a specific connection point and the number of connection points can be adjusted by the user. The new parameter estimation block, called TUNE is presented in Figure 19 and an example implementation with a heat exchanger model can be seen from Figure 20.

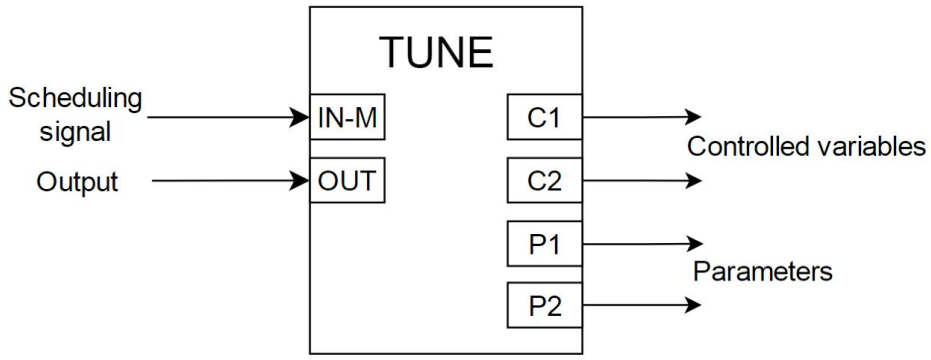


Figure 19. The parameter estimation block, TUNE.

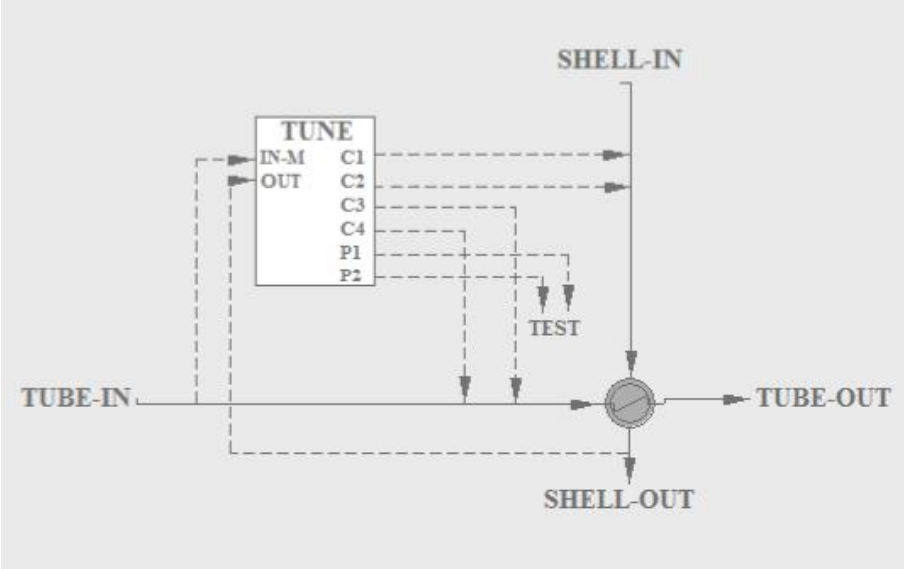


Figure 20. Example configuration of the TUNE block.

The parameter tuning block has three modes; OFF, TUNE and OPTIM. In the OFF mode the block does not affect the model. In the OPTIM mode the parameters are estimated and in the TUNE mode the parameters are switched according to the parameter transition schedule. The functionality of the parameter transition schedule is illustrated in Figure 21.

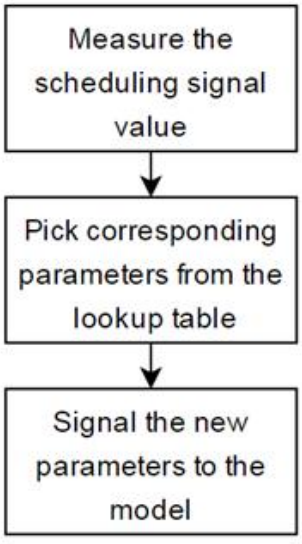


Figure 21. The parameter transition functionality.

The user interface of the parameter estimation tool was implemented and connected to the parameter estimation block. Data can be imported to the parameter estimation tool with a data file. The tags in the data file are used to connect the measurements to the corresponding signals. User can also easily check the configuration of the signals and give upper and lower parameter limits, as well as the initial values for the parameters. The user interface also has the optimization method selection and initialization options. In addition the user can test the objective function sensitivity to the selected parameters by estimating the gradient in the initial point with the finite-differences method. The interface is presented in Figure 22.

During the parameter estimation, user can monitor the progress from a specific chart showing the critical measurements and the residual as seen from Figures 23 and 24. Finally the results of the parameter estimation are shown in the user interface as well as the final residual and the number of needed iterations.

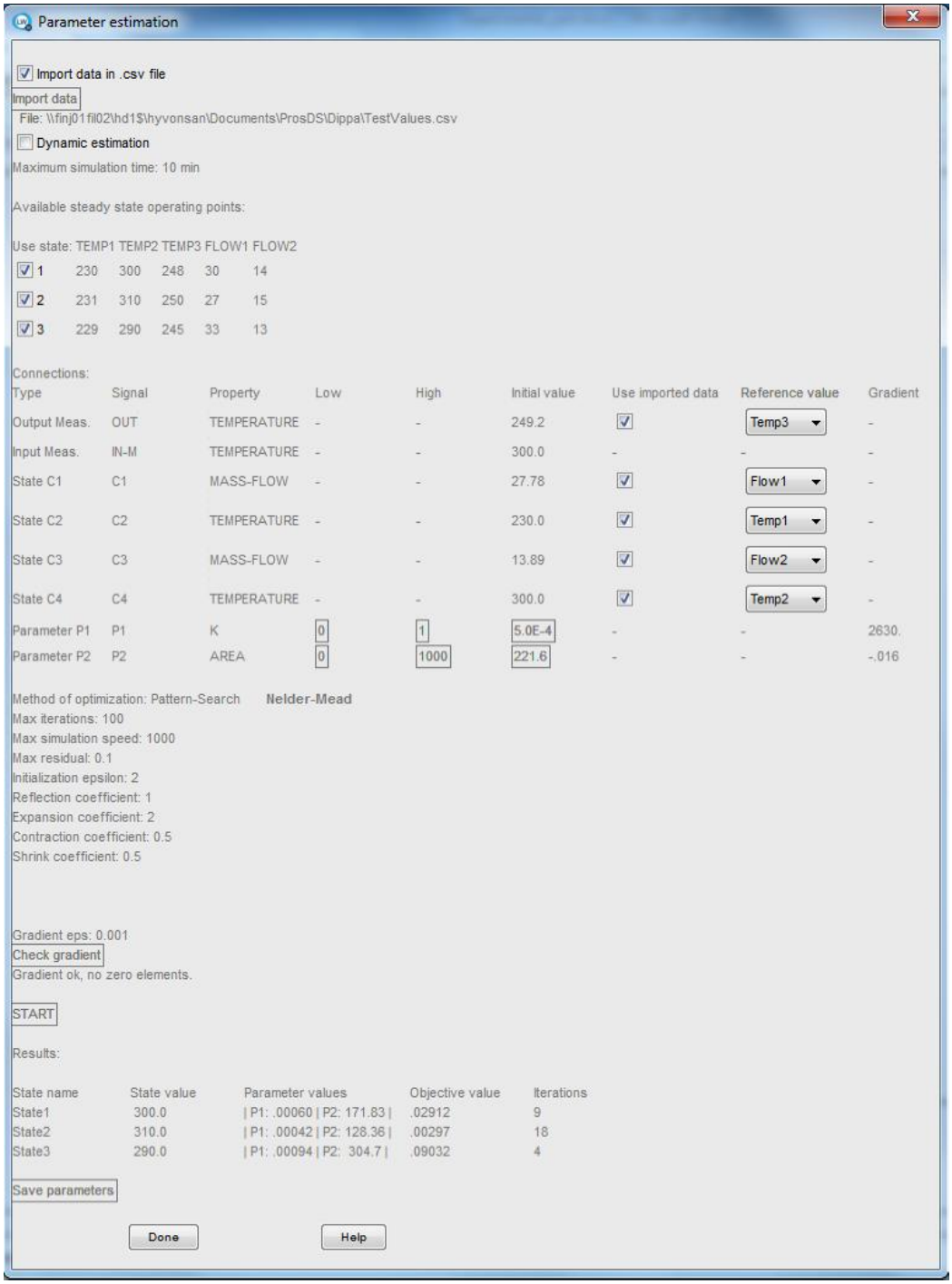


Figure 22. User interface of the parameter estimation tool.

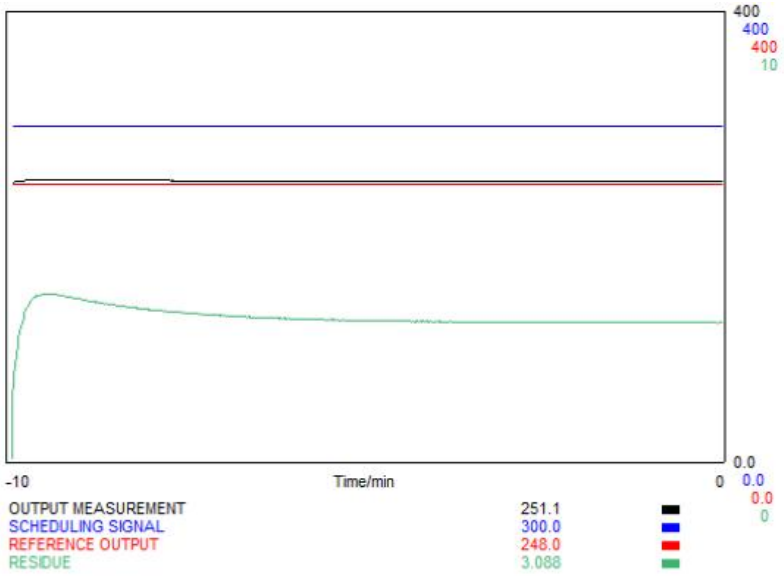


Figure 23. Initial guess for the parameter values.

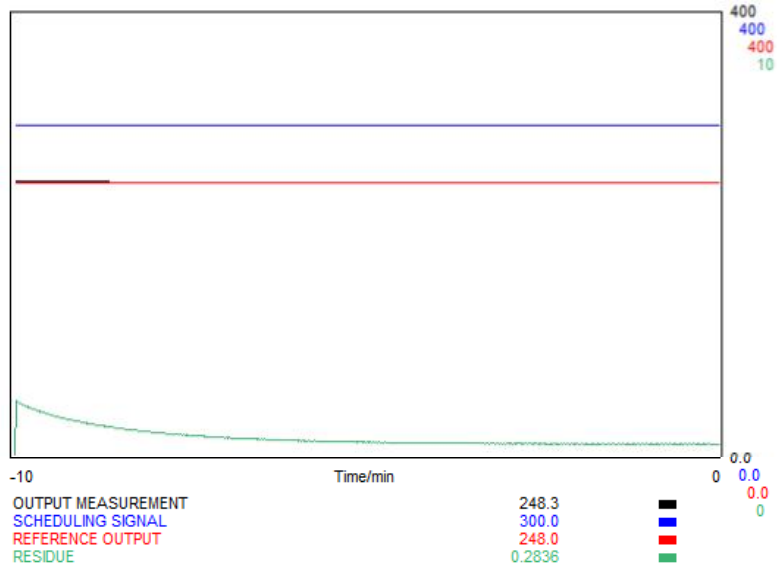


Figure 24. Improved guess for the parameter values.

9. Testing

The designed and implemented methodology was tested with process data collected from an industrial process to verify the applicability of the methods. In this chapter first the test unit is described and then the test arrangements are discussed.

9.1 Description of the test process unit

Heat exchangers are widely used process equipment in process industry and the dynamics of heat transfer are well understood. However, when heat exchangers are modeled as a part of a large process model, usually only simplified models can be used due to limited processing capacity and simulation speed requirements. In addition, some of the important heat exchanger parameters, such as the heat transfer coefficient can be difficult to define exactly.

The overall heat transfer coefficient depends on a number of different variables, such as the composition of the flows, the heat exchanger type and material, phases of the fluids and turbulence of the flows. Measurement of all relevant variables such as the exact composition, in industrial applications, is either impossible or expensive. Another aspect is that the heat transfer coefficient has also time-varying and non-linear behavior, due to fouling or phase-changes inside the heat exchanger.

The NAPCON ProsDS dynamic process simulator has different heat exchanger models implemented as default. However, these models are simplified and are often only tuned to model one specific narrow operating regime or a specific operating point. As described previously, the tuning of the model parameters is currently a manual task. Due to the large number of heat exchangers in process models, the parameter estimation of heat exchangers takes a significant amount of time.

Due to these factors, a heat exchanger model was selected for testing the parametrization tool. The selected countercurrent heat exchanger had been previously modeled as a part of an operator training simulator model. In the process, the heat exchanger is a part of a heat integration heat exchanger train that contains total of eight heat exchangers. The heat exchanger train preheats the product flow from a product stripper before it enters a fractionator column. Both flow measurements and all

temperature measurements were available for the unit. The available measurements have been described in Figure 25.

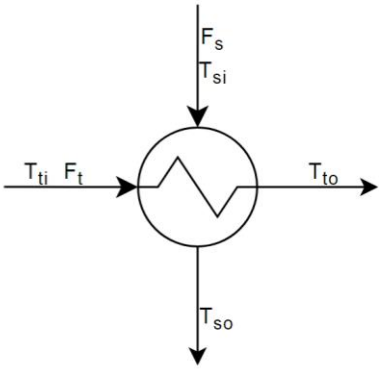


Figure 25. Available measurements from the selected heat exchanger. The subscript letter s represents shell side and the t represents tube side.

The NAPCON ProsDS heat exchanger model consists of two stirred tanks, tube and shell sides. The heat transfer between the tanks is calculated as in equation 29. The simple model does not take into account the temperature gradient in the heat exchanger; instead the heat exchange is calculated as changing heat between two fluids in homogeneous temperature. Therefore, the simple heat exchanger model does not model a countercurrent heat exchanger properly and the maximum achieved output temperatures are equal, when the temperature difference is zero. The simple model diagram can be seen from Figure 26.

$$q = kA\Delta T * b_{corr} \tag{29}$$

Where q represents the heat transfer, A is the heat transfer area, k is the heat transfer coefficient, ΔT is the temperature difference and b_{corr} is a correction coefficient which takes into account for example the liquid level in the tanks.

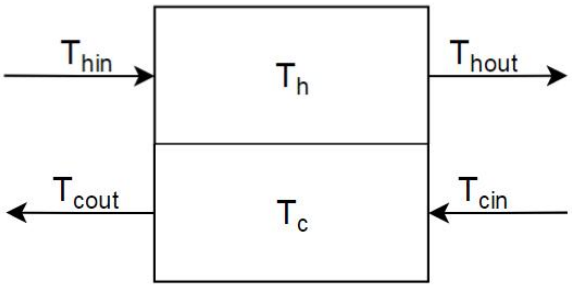


Figure 26. A heat exchanger modeled with one calculation element, adapted from [88]. The subscript letter h represents hot flow and the c represents the cold flow.

A hypothesis was made, that the model match with the data could be improved by splitting the heat exchanger model to several calculation elements similarly as in the finite elements method. It was chosen that three calculation elements would be used to give the model enough complexity to match the data, but in the same time keep the simulation speed reasonable. The diagram for heat exchanger modeled with three calculation elements can be seen from Figure 27.

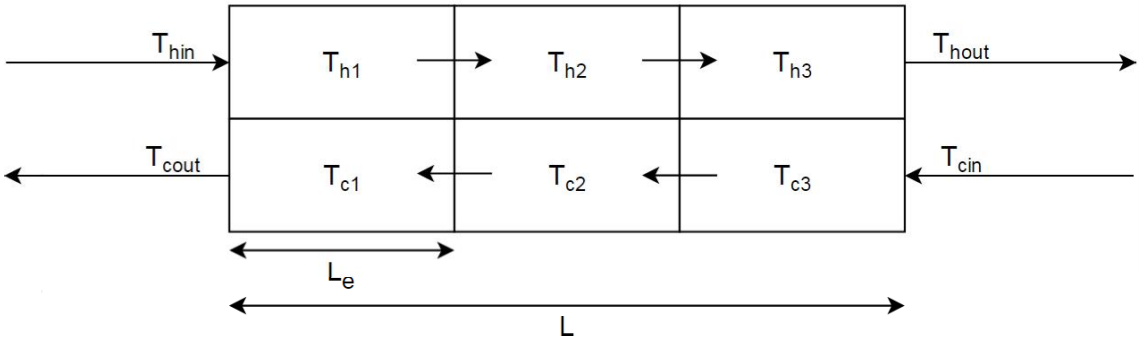


Figure 27. A heat exchanger modeled with three finite calculation elements, adapted from [88].

9.2 Test arrangements

The purpose of the data partition test was to test the available data visualization and clustering algorithms for identifying steady state operating points from real process data. Measurement data was collected from the process from five months period. By visually inspecting the data, an outlier point was detected and marked as missing data. After preprocessing, the data was normalized and visualized using histograms and scatter matrix in the Scilab program. Then the selected clustering algorithms were tested on the data. After the data was partitioned, the mean values of the selected clusters were calculated and printed to file. This was used as the data file for the parameter estimation test.

The aim of the parameter estimation test was to apply the developed parameter estimation tool for tuning a dynamic heat exchanger model to the identified steady state operating states. As mentioned in the previous subchapter, the selected heat exchanger unit had been previously modeled as a part of an operator training simulator. This model

was selected as the starting point for parameter estimation. Contents of the flows as well as pressures were calculated as a part of the operator training simulator model.

Two test models were built, first one that contained only one calculation element, and a second one that had the calculation split into three finite calculation elements. The parameter estimation was executed with both the Nelder-Mead and the pattern search methods for both models. For the second model the heat transfer coefficient values were allowed to change between calculation units to allow more flexibility in the model and therefore three parameters were estimated. For the Nelder-Mead method the key optimization parameters can be seen from Table 2 and for pattern search from Table 3. After the parameter estimation, the parameters were saved to a lookup table based on the selected scheduling variable and the models were tested in the identified operating states.

Table 2. Nelder-Mead optimization parameters.

Initialization parameter ε	Reflection coefficient	Expansion coefficient	Contraction coefficient	Shrink coefficient
2	1	2	0.5	0.5

Table 3. Pattern search optimization parameters.

Initial step size	Step limit
0.001	1.0E-7

10. Results and discussion

As a continuation to the previous chapter, the achieved test results are presented and discussed in this chapter. First subchapter presents the results. Then, the results are evaluated and discussed in the second subchapter. Finally, in the last subchapter, recommendations based on the results as well as suggestions for continuing this study are made.

10.1 Results

10.1.1 Data partition to operating points

The measurements collected from the real process can be seen from Figure 28. By visually inspecting the data, two operating points were identified; the normal operating point and a second, temporary operating point. Based on the visualization, temperature and flow into the tube side were selected as clustering variables. Results of the visualization for selected variables can be seen from Figure 29 and for all variables from Appendix 1.

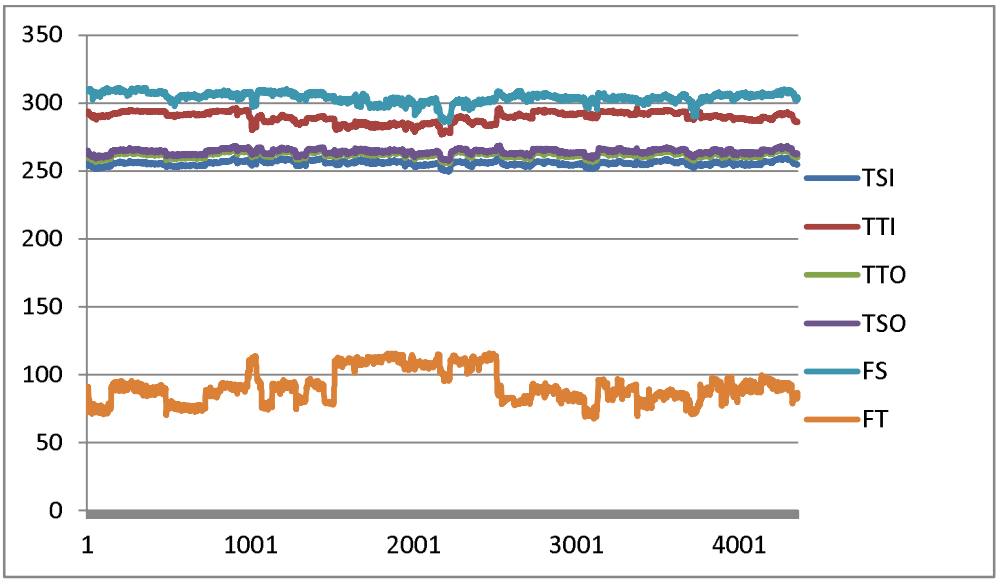


Figure 28. Preprocessed measurement data.

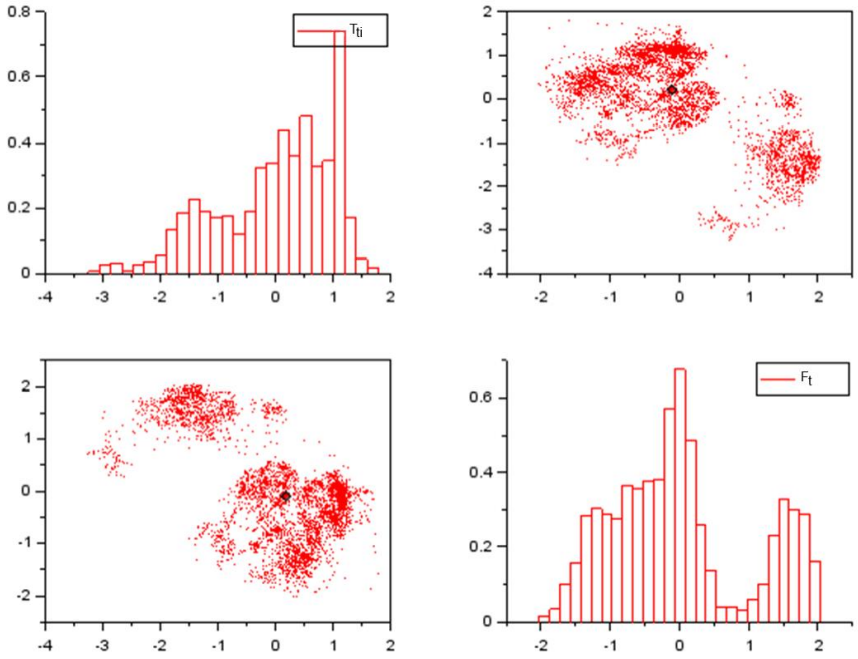


Figure 29. A Scatter matrix and histograms for selected variables.

From Figure 29 two clusters were visually identified and in addition the histograms indicated existence of two separate clusters. From these, the larger cluster represents the normal operating area, when the smaller cluster represents the abnormal operating state. After visualization, three clustering algorithms were tested on the data with different clustering parameters. Results of different algorithms and different parameters can be seen from Appendix 2. Based on visual inspection of the results, the DBSCAN algorithm, with clustering parameters distance 0.1 and minPts 30, was selected as the best alternative. Results with the selected algorithm and clustering parameters can be seen from Figure 30.

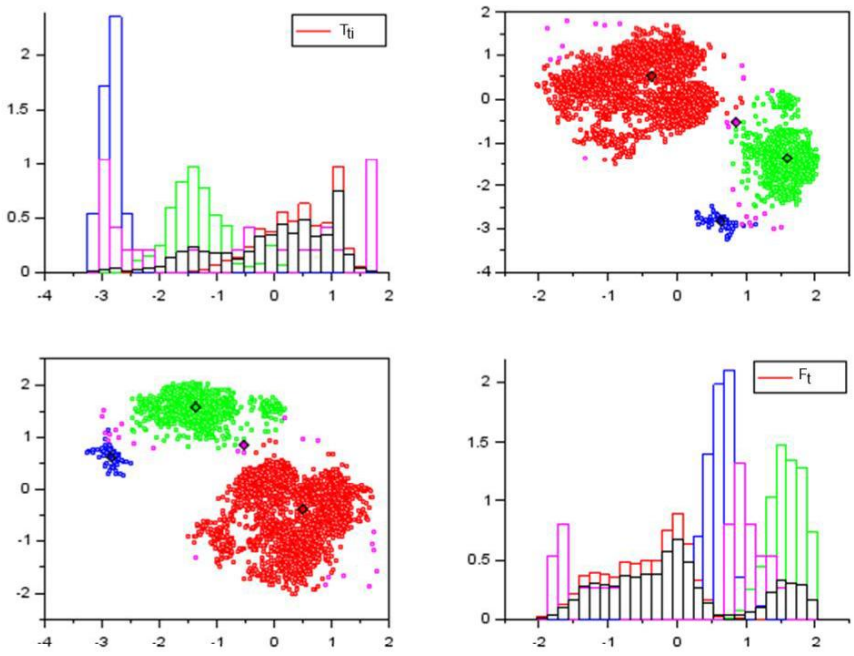


Figure 30. The clustering result with DBSCAN with distance of 0.1 and minPts of 30.

In Figure 30, the red cluster represents the normal operating point and the green the temporal operating point. Based on the visual inspection, clusters 1 (red) and 2 (green) were selected because they seemed to contain the interesting operating point data. Mean values of the selected operating points can be seen from Table 4.

Table 4. Operating states based on data clustering.

Operating point	T_{si} °C	T_{ti} °C	T_{to} °C	T_{so} °C	F_s kg/s	F_t kg/s
1	255.9	291.4	262.2	264.4	84.9	23.9
2	256.0	284.6	262.5	264.3	83.6	30.7

10.1.2 Parameter estimation

The heat transfer coefficient was selected as the parameter for estimation and the shell side output was selected as the objective variable for model tuning. Based on the plotted data, the flow to the tube side was selected as the scheduling variable. Initial guess for the heat transfer coefficient was 500 $\mathrm{W/m^2K}$. For both models, a smaller objective value was achieved with the Nelder-Mead method and a better match with data was achieved with the second model. Summary of the parameter estimation results can be

seen from Tables 5 and 6. The following Figures show trends from the simulation tests for the second model.

Table 5. The results of parameter estimation for model with one calculation unit.

Method	Estimation time *s*	Function evaluations	Operating point	Parameter k value $\mathrm{kW/m^2K}$	Objective value
Nelder-Mead	360	488	1	159.8	0.77
			2	171.5	0.81
Pattern search	76	50	1	10.5	0.93
			2	11.5	0.99

Table 6. The results of parameter estimation for model with three calculation units.

Method	Estimation time *s*	Function evaluations	Operating point	Parameter values $\mathrm{kW/m^2K}$			Objective value
				k1	k2	k3	
Nelder-Mead	27	23	1	1.5	1.5	1.5	0.040
			2	2.4	1.4	1.2	0.030
Pattern search	40	44	1	1.5	1.5	1.5	0.040
			2	1.5	1.5	2.5	0.077

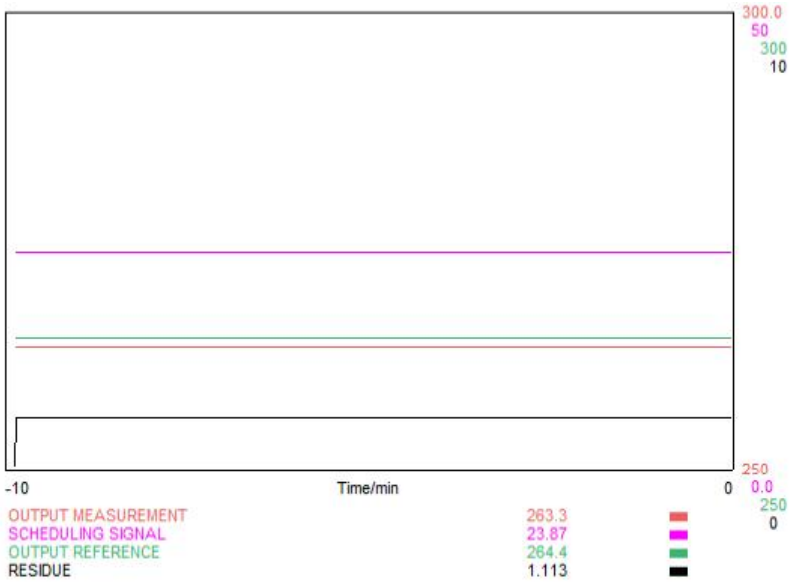


Figure 31. A simulation test with the initial parameter values for the first operating state.

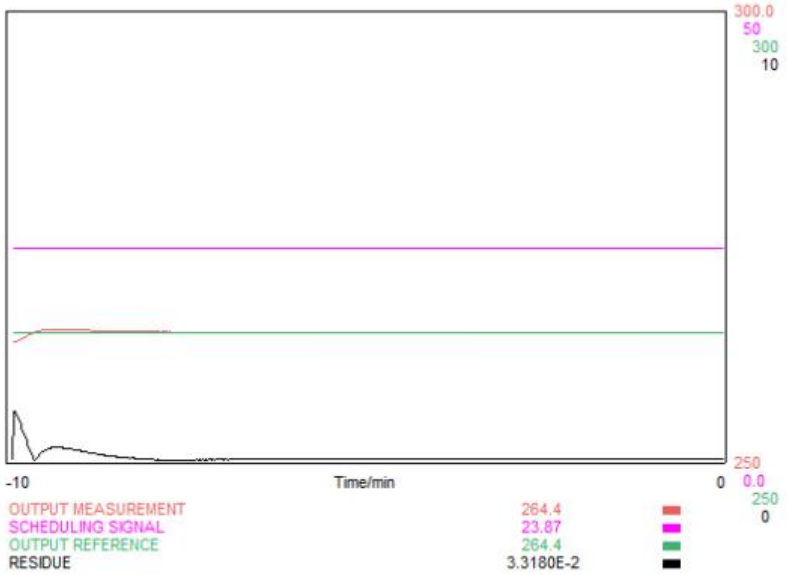


Figure 32. A simulation test with the final estimated parameter values for the first operating state.

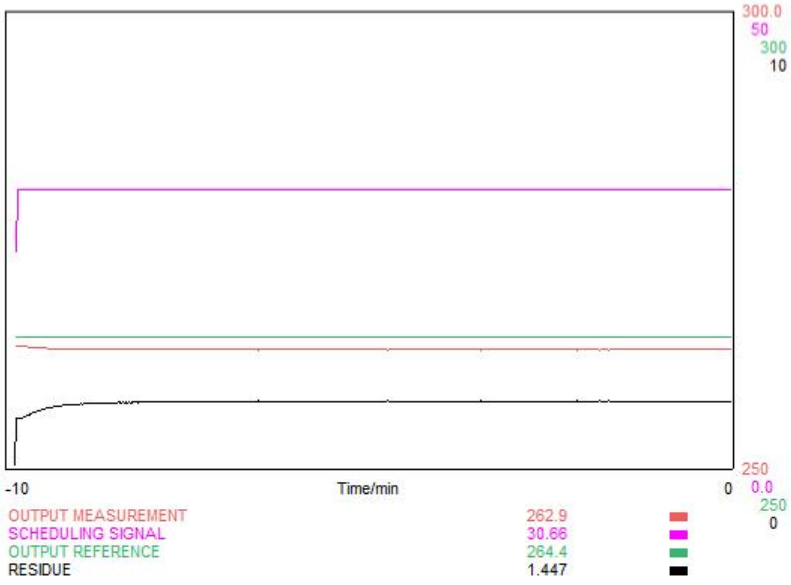


Figure 33. A simulation test with the initial parameter values for the second operating state.

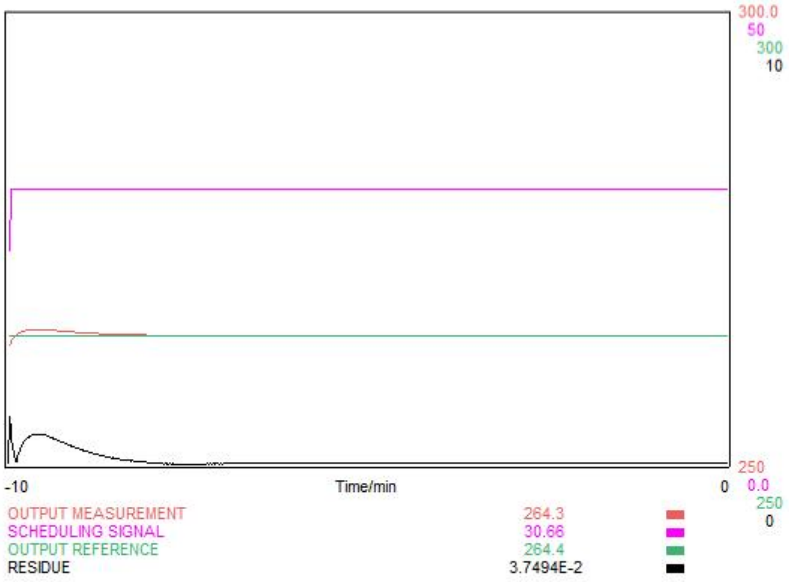


Figure 34. A simulation test with the final estimated parameter values for the second operating state.

Based on the parameter estimation, results with the Nelder-Mead algorithm were selected for further testing. Results for model behavior with the estimated parameter schedule for both operating states can be seen from Tables 7 and 8.

Table 7. Model output measurements before and after the parameter estimation for the model with one calculating element.

Operating point	Measurement	Reference value	Before parameter estimation	After parameter estimation
1	T_{so} / °C	264.4	261.1	263.6
	T_{to} /°C	262.2	273.1	263.7
2	T_{so} / °C	264.4	260.7	263.6
	T_{to} /°C	262.5	271.6	263.6

Table 8. Model output measurements before and after the parameter estimation for the model with three calculating elements.

Operating point	Measurement	Reference value	Before parameter estimation	After parameter estimation
1	T_{so} / °C	264.4	262.3	264.4
	T_{to} /°C	262.2	268.6	260.4
2	T_{so} / °C	264.4	261.9	264.3
	T_{to} /°C	262.5	268.5	261.2

10.2 Discussion on results

10.2.1 Data partition

The partition of the process data to different operating points was successfully conducted with the DBSCAN algorithm. As discussed in the theoretical part, the algorithm is able to reject possible outliers in the data. In the clustering sense, this was implemented as stricter rules for classifying the data. For more strict definition of which data points belonged to the same cluster, the temporal fluctuations were restricted out of the cluster. This was beneficial because in this thesis the aim was to identify stable, long term steady state points and reject the temporal shifts. However, if the aim would have been to estimate parameters for dynamic behavior, then too strict clustering would lead to the loss of relevant data.

The validity of the selected clustering result was evaluated numerically and by comparison to the other methods. To evaluate the result numerically, the incidence matrix, that describes whether or not data points belong to the same cluster, and the distance matrix, that describes the distance between two points, were calculated. Then the correlation coefficient of these two matrixes was calculated. For the selected cluster result was -0.396. This is decent result since the evaluation method is not optimal for density based clusters. In addition the correlation coefficient for the selected method was better than what was achieved with 2-3 clusters with the kernel k-means or the k-means++. The best results for the selected methods can be seen from Table 9.

Table 9. Numerical evaluation of the clustering results.

Method	Correlation coefficient
DBSCAN	-0.396
Kernel k-Means	-0.250
K-means++	-0.392

The cluster validity was also verified by visually comparing the result to results with different clustering parameters and different clustering methods. The selected result had better formed clusters and the mixing of different clusters was lower. As can be seen from Appendix 2, the results with the kernel k-means and k-means++ suffered from the tendency of these methods to form uniformly sized clusters. On the other hand, the DBSCAN method with a good density-level was able to form clusters with different shapes and sizes, as was expected.

However, the selection of the density-level required trial and error style testing. As seen from Appendix 2, with wrong clustering parameters the clusters are either merged to one cluster or too many clusters are formed. Based on the literature survey, this could be avoided by using a clustering method with a varying density-level.

Finally the clustering result was compared with the plotted data to determine if a non-random structure actually exists in the data. The second operating point, based on the clustering, was highlighted from the plot, as seen from Figure 35. As seen from the figure, the second operating point was identified from the data, but also an additional process shift was included in the second cluster. However, the plot implies that the clusters were formed on actual operating points rather than random structures in the data.

Based on these initial results, the selected DBSCAN algorithm seems to be a promising clustering analysis tool for partitioning the process data space to operating points. In addition, the selected visualization methods, the scatter matrix and the histograms, seem to be promising tools for the modeler to better understand the data. However, only very limited testing was conducted and more testing is required to verify the results.

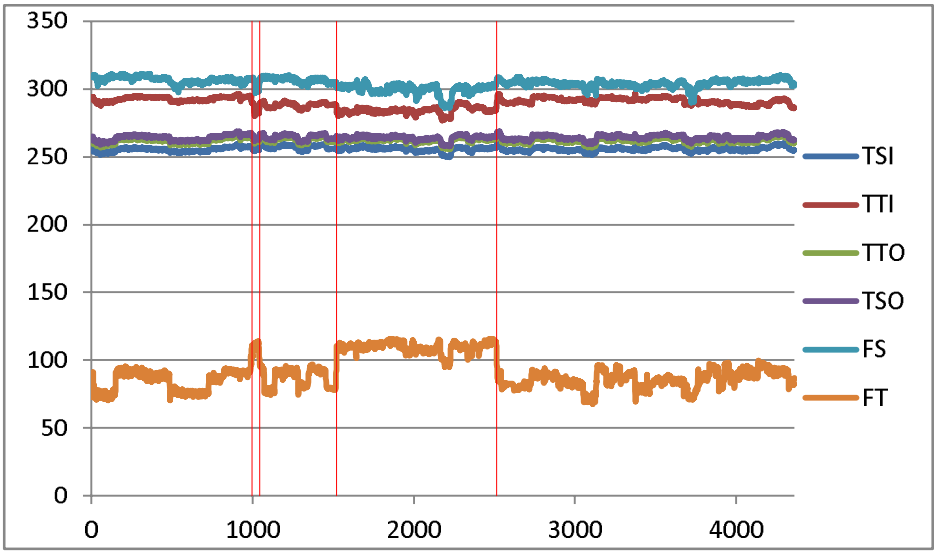


Figure 35. The second identified operating point highlighted from the data.

10.2.2 Parameter estimation

The aim of the parameter estimation tool was to offer a general solution for different kinds of parameter estimation problems. This was achieved by implementing the tool as a separate block in the simulator. However, the generality of the tool makes it harder for the user to configure. In the future, the functionality could be implemented as a model object specific feature that would make it easier for the user to configure.

The user interface was built in a way that, in the typical case, the user only has to select the data and match the signals with corresponding tags. In addition, the gradient estimation gives the user information about the sensitivity of the model to different parameters. This should make the configuration easier. However, some options, such as the needed simulation time before model reaches a new steady state, depend heavily on the model complicity and dynamics. These options the user has to estimate based on process knowledge and simulation tests.

The selected optimization methods were robust and easy to use, but due to their simplicity likely to converge slower than more advanced methods as was discussed in the theoretical part. However, even with the more complex model the convergence was achieved in a reasonable time. In addition, the first model had major oscillations when the temperature difference was small. This would probably have caused problems with

the gradient optimization methods. The main criterion of automating the previously manual parameter estimation process was met.

Two models were tested with the parameter estimation tool for the same set of data. The first model was very simple and, as discussed in the previous chapter, it could not be tuned to represent the data exactly. As was discussed, the theoretical limit was equal output temperatures, and this was also seen in the parameter estimation as the temperatures got closer, the effect of change in parameters got lower. This caused both of the optimization methods to overestimate the parameters. To improve this, more advanced stopping rules could be used that would take the current rate of objective change into account. However, if only the model output is of interest, not the physical reliability of parameters, then the results of the parameter estimation are acceptable as better fit to the data was achieved.

As was assumed in the previous chapter, the heat exchanger model that was split to three finite elements was able to match the data better. The crosscurrent heat exchanger heat transfer was modeled correctly and the fit to the data was significantly improved with the parameter estimation. In addition the estimated model parameters were physically reasonable.

The parameter scheduling lookup table worked as was intended and was able to change the parameters to correspond with the changing scheduling variable. The flow to the tube side was selected as the scheduling variable, because it had clearly different values in the two operating states. Therefore, the estimated parameters were inserted to the lookup table as the function of the tube side flow.

The selection of the scheduling variable is no minor task because it should represent the changing of the operating point. However, one operating state can depend on numerous different variables. The simple lookup table was still an improvement compared to a model only covering one operating state or having to manually code the parameter transition schedule. In addition, the lookup table structure makes the changing of the parameters transparent and easy to present for the user.

Here the focus was on the data based approach and the reason behind the temporal operating point was not investigated. For gaining better understanding of the process

dynamics and making more accurate models the modeler should have understanding of the reasons affecting the current process state. However this was out of the scope of this thesis.

The Nelder-Mead algorithm is a promising optimization method for the parameter estimation tool, because, as was discussed, it requires less simulation tests per iteration than the pattern search method. This was seen in the parameter estimation of the second model when the method had faster convergence. However, for non-converging parameter estimation, like the first model, the stopping rules did not stop the iteration and this caused the estimation to run to the maximum iterations. To improve this, better stopping rules are needed for the tool.

Based on the results, the implemented parameter estimation tool seems to be promising for the estimation of simulator model parameters. However, the tool was only tested with steady state data and only for one test case. More testing with different types of process models is required to proof the applicability of the tool.

The achieved results imply that the developed methodology successfully met the set requirements. By applying the methodology, a dynamic simulator test model was semi-automatically tuned to better correspond to plant data with several operating states. However, more testing of the implemented methodology is required before the general applicability can be verified.

10.3 Further study

The aim of this thesis was to study and test methods for a new parameter estimation methodology. Therefore this thesis has not covered detailed study of all the used methods; rather a short overview of the most interesting methods and technologies has been presented. This thesis has crossed academic borders by studying methods in the field of model identification as well as in the field of machine learning. Both fields enjoy the active interest of the academic community and therefore in the future the methodology could be improved by making a more detailed study focusing on either of these subjects.

Hundreds of different machine learning algorithms have been presented in literature. This thesis covers just a few algorithms and tests are conducted only for one set of data. Interesting subject for future continuation of this work would be the application of clustering algorithms to large databases. Especially the presented newer density-based algorithms could be promising for their suitability to clustering large data sets and better adjusting to different density levels.

In this thesis the Scilab software was used for testing the selected machine learning algorithms. However, these algorithms could be easily implemented as a part of the NAPCON Information Manager software. This would enable direct data collection from the database and automated application of the machine learning algorithms.

This thesis was focused on finding steady state operating points from the data and then estimating parameters for dynamic models at these specific states. Therefore dynamic parameter estimation was not covered here. The parameter estimation of dynamic simulator models in dynamic situations is left for future research. Another interesting aspect is that here only one output variable was selected as the objective, when often the modeler is interested about optimizing several variables simultaneously. Therefore another interesting area for future research is multiobjective parameter estimation for simulator models.

The selected parameter estimation method was implemented as a new tool in the NAPCON ProsDS simulator. However the tool only offers two optimization algorithms. More advanced algorithms could be implemented such as the gradient methods to improve convergence speed in some situations. In addition the usability of the tool could be improved by implementing it as a functionality of the process model blocks.

11. Conclusion

Process simulators are commonly used modeling tools in the process industry and their applications include process design, safety evaluations and operator training. However, the modeling of real processes is hardly a trivial task and new tools are required to improve the modeling efficiency as well as the reliability of the models.

The area of data mining has been the subject of growing academic and industrial interest during the last years. The advancements in information technology, data acquisition systems and storage technology as well as development in machine learning tools have created the interest for applying data mining tools for automatic model configuration and tuning.

Before this thesis, no known methodology for combining operating point identification with parameter estimation of simulator models had existed. The aim of the thesis was to develop a new methodology for applying data mining analysis together with parameter estimation for improving simulator model match to data.

In the theoretical part, a literature survey of clustering methods suitable for operating state identification was presented. The survey presented commonly used clustering algorithms and in addition several recent studies were introduced. Then classical parameter estimation methods were presented as well as a survey of iterative parameter estimation methods and suitable general purpose optimization methods. Then a review of state conscious identification strategies was made. Finally, as the basis for the methodology development, several examples of parameter estimation applications in simulators were presented.

In the experimental part, the current state of the parameter estimation process was discussed and objectives for the methodology were formulated. Based on the literature survey, the methods for the operating space partition, parameter estimation and for the parameter scheduling were selected.

The methodology was then implemented and tested by tuning an operator training simulator model to several operating points based on plant process data. Significant improvement was achieved in the model behavior by increasing the model complicity and then estimating the model parameters.

Based on the results achieved, it was concluded that the developed methodology was able to tune the selected heat exchanger model to better match the data in the identified operating points. However, more testing with different processes and different data sets is required to verify the general applicability of the methodology.

References

[1] I. Cameron and R. Gani, Product and process modelling: a case study approach, 1 ed., Amsterdam: Elsevier, 2011, p. 555.

[2] J. A. Harding, M. Shahbaz and A. Kusiak, "Data Mining in Manufacturing:," *Journal of Manufacturing Science and Engineering,* vol. 128, no. 4, pp. 969-976, 2015.

[3] G. Govaert, Data Analysis, 1 ed., G. Govaert, Ed., Hoboken: Wiley, 2010, p. 343.

[4] M. Majid, S. Sulaiman, H. Mokhtar and A. Tamiru, "Operating point estimation for an absorption process using data clustering technique," *Journal of Applied Sciences,* vol. 13, no. 3, pp. 377-384, 2013.

[5] A. A. Adebowale and S. El Ferik, "Modeling and Identification of Nonlinear Systems: A Review of the Multimodel Approach - Part 1," *IEEE Transactions On Systems, Man, and Cybernetics: Systems 2016,* pp. 1-11, 2016.

[6] R. Isermann and M. Münchhof, Identification of Dynamic Systems: An Introduction With Applications, Berlin: Springer, 2011, p. 705.

[7] R. Srinivasan, "Artificial intelligence methodologies for agile refining: an overview," *Knowledge and Information Systems,* vol. 12, no. 2, pp. 129-145, 2007.

[8] R. Srinivasan, C. Wang, W. K. Ho and K. W. Lim, "Dynamic PCA based methodology for clustering process states in agile chemical plants," *Industrial and Engineering Chemistry Research,* vol. 43, no. 9, p. 2123–2139, 2004.

[9] V. Estivill-Castro, "Why so many clustering algorithms: a position paper," *ACM SIGKDD Explorations Newsletter,* vol. 4, no. 1, pp. 65-75, 2002.

[10] R. Sibson, "SLINK: an optimally efficient algorithm for the single-link cluster method," *The Computer Journal,* vol. 16, no. 1, pp. 30-34, 1973.

[11] D. Defays, "An efficient algorithm for a complete-link method," *The Computer Journal,* vol. 20, no. 4, pp. 364-366, 1977.

[12] S. Lloyd, "Least squares quantization in PCM," *Telephone Labs Memorandum, Murray Hill, NJ, reprinted (1982) in IEEE Trans. Information Theory,* vol. 2, pp. 129-137, 1957.

[13] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, New Orleans, 2007.

[14] J. Bezdek, "Pattern Recognition with Fuzzy Objective Function Algorithms," Plenum Press, New York, 1981.

[15] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids," in *Statistical data analysis based on the L1-norm and related methods*, Amsterdam, North-Holland, 1987, p. 464.

[16] L. Kaufman and P. Rousseeuw, Finding Groups in Data, New York: Wiley, 1990.

[17] R. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *VLDB'94, Proceedings of 20th International Conference of Very Large Data Bases*, Santiago de Chile, 1994.

[18] R. Hogg, J. McKean and A. Craig, Introduction to Mathematical Statistics, Harlow: Pearson Education, 2014.

[19] X. Xu, M. Ester, H.-P. Kriegel and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *Proceedings of 14th International Conference on Data Engineering*, Orlando, 1998.

[20] H. Kriegel, P. Kröger, J. Sander and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews-Data Mining And Knowledge Discovery,* vol. 1, no. 3, pp. 231-240, 2011.

[21] M. Ester, H. Kriegel, J. Sander and X. Xu, "A Density-Based Algorithm for Discovering Clustersin Large Spatial Databases with Noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Portland, 1996.

[22] A. Hinneburg and D. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proceedings of the 4th ACM International Conference on Knowledge Discovery and Data Mining*, New York, 1998.

[23] M. Ankerst, M. Breunig, H.-P. Kriegel and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *Proceedings of the ACM International Conference on Management of Data*, Philadelfia, 1999.

[24] L. Ertöz, M. Steinbach and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proceedings of the 3rd SIAM*

International Conference of Data Mining, San Francisco, 2003.

[25] T. Pei, A. Jasra, H. DJ., A.-X. Zhu and C. Zhou, "DECODE: a new method for discovering clusters of different densities in spatial data," *Data Mining Knowledge Discovery,* vol. 18, pp. 337-369, 2009.

[26] M. Ronen, Y. Shabtai and H. Guterman, "Hybrid model building methodology using unsupervised fuzzy clustering and supervised neural networks," *Biotechnology and Bioengineering,* vol. 77, no. 4, p. 420–429, 2002.

[27] N. D. J.-Y. B. M. B. P. Elfelly, "A new approach for multimodel identification of complex systems based on both neural and fuzzy clustering algorithms," *Engineering Applications of Artificial Intelligence,* vol. 23, no. 7, p. 1064–1071, 2010.

[28] N. Elfelly, J.-Y. Dieulot and P. Borne, "A neural approach of multimodel representation of complex processes," *International Journal of Computers, Communications and Control,* vol. 3, no. 2, pp. 149-160, 2008.

[29] B. Hartmann, N. O. I. Skrjanc and A. Sodja, "Supervised hierarchical clustering (SUHICLUST) for nonlinear system identification," in *IEEE Symposium on Computational Intelligence in Control and Automation*, Nashville, 2009.

[30] L. Teslic, H. B. O. Nelles and I. Skrjanc, "Nonlinear system identification by Gustafson–Kessel fuzzy clustering and supervised local model network learning for the drug absorption spectra process," *IEEE Transactions on Neural Networks,* vol. 22, no. 12, pp. 1941-1951, 2011.

[31] J. Abonyi, R. Babuska and F. Szeifert, "Modified Gath-Geva fuzzy clustering for identification of Takagi-Sugeno fuzzy models," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 32, no. 5, pp. 612-621, 2002.

[32] T. Kohonen, "Self organized formation of topological correct feature maps," *Biological Cybernetics,* vol. 43, pp. 59-69, 1982.

[33] S.-L. Jämsä-Jounela, M. Vermasvuori, S. Haavisto and P. Endén, "A toolbox for on-line process monitoring with some industrial applications," in *b´02 Barcelona 2002 IFAC 15th World Congress*, Barcelona, 2002.

[34] L. Ljung, System Identification: Theory for the User, 2 ed., Upper Saddle River: Prentice Hall, 1999, p. 609.

[35] B. A. Ogunnaike and W. H. Ray, Process Dynamics, Modeling, and Control, New

York: Oxford U.P, 1994, p. 1260.

[36] N. Kalogerakis and P. Englezos, Applied Parameter Estimation for Chemical Engineers, New York: CRC Press, 2001, p. 459.

[37] R. Doraiswami, M. Stevenson and C. Diduch, Identification of physical systems: Applications to condition monitoring, fault diagnosis, softsensor, and controller design, 1 ed., Chichester, West Sussex: John Wiley & Sons, 2014, p. 538.

[38] J. Nocedal and S. Wright, Numerical Optimization, New York: Springer, 1999.

[39] F. Potra and S. Wright, "Interior-point methods," *Journal of Computational and Applied Mathematics,* vol. 124, no. 1-2, pp. 281-302, 2000.

[40] A. Gosavi, Simulation-based optimization: Parametric optimization tehniques and reinforcement learning, 2 ed., Boston: Springer, 2014.

[41] T. Kolda and R. T. V. Lewis, "Optimization by Direct Search; New Perspectives on Some Classical and Modern Methods," *SIAM Review,* vol. 45, no. 3, pp. 385-482, 2003.

[42] R. Hooke and T. A. Jeeves, ""Direct search" solution of numerical and statistical problems," *Journal of the Association for Computing Machinery (ACM),* vol. 8, no. 2, pp. 212-229, 1961.

[43] W. C. Davidon, "Variable metric method for minimization," *SIAM Journal on Optimization,* vol. 1, pp. 1-17, 1991.

[44] G. E. P. Box, "Evolutionary Operation: A method for increasing industrial productivity," *Applied Statisctics,* vol. 6, pp. 81-101, 1957.

[45] J. E. Dennis and V. Torczon, "Direct search methods on parallel machines," *SIAM Journal on Optimization,* vol. 1, pp. 448-474, 1991.

[46] V. J. Torczon, "On the convergence of pattern search algorithms," *SIAM Journal on Optimization,* vol. 7, no. 1, pp. 1-25, 1997.

[47] J. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal,* vol. 7, pp. 308-313, 1965.

[48] R. Luus and T. H. I. Jaakola, "Optimization by direct search and systematic reduction of the size of search region," *AIChE Journal,* vol. 19, p. 760–766, 1973.

[49] R. Luus, "Determination of the Region Sizes for LJ Optimization Procedure,"

Hungarian Journal of Industrial Chemistry , vol. 26, no. 4, pp. 281-286, 1998.

[50] R. Srinivasan, P. Viswanathan, H. Vedam and A. Nochur, "A framework for managing transitions in chemical plants," *Computers & Chemical Engineering,* vol. 29, no. 2, pp. 305-322, 2005.

[51] J. J. Peyton and K. Muske, "Identification and adaptation of linear look-up table parameters using an efficient recursive least-squares technique," *ISA Transactions,* vol. 48, no. 4, pp. 476-483, 2009.

[52] M. Vogt, N. Muller and I. R., "On-Line Adaptation of Grid-Based Look-up Tables Using a Fast Linear Regression Technique," *Journal of Dynamic Systems, Measurement, and Control,* vol. 126, no. 4, pp. 732-739, 2005.

[53] W. Rugh and J. S. Shamma, "Research on gain scheduling," *Automatica,* vol. 36, no. 10, p. 1401–1425, 2000.

[54] A. A. Bachnas, R. Tóth, J. H. A. Ludlage and A. Mesbah, "A review on data-driven linear parameter-varying modeling approaches: A high-purity distillation column case study," *Journal of Process Control,* vol. 24, no. 4, pp. 272-285, 2014.

[55] J. Huang, G. Ji and Y. Zhu, "Some study on the identification of multi-model LPV models with two scheduling variables," in *16th IFAC Symposiumon System Identification*, Brussels, 2012.

[56] J. Huang, G. Ji, Y. Zhu and P. van den Bosch, "Identification of multi-model LPV models with two scheduling variables," *Journal of Process Control,* vol. 22, no. 7, p. 1198–1208, 2012.

[57] X. Jin, B. Huang and D. S. Shook, "Multiple model LPV approach to nonlinear process identification with EM algorithm," *Journal of Process Control,* vol. 21, no. 1, p. 182–193, 2011.

[58] R. Tóth, P. M. J. Van den Hof, J. H. A. Ludlage and P. S. C. Heuberger, "Identification of nonlinear process models in an LPV framework," in *9th International Symposium on Dynamics and Control of Process Systems*, Leuven, 2013.

[59] K. Hsu, T. L. Vincent and K. Poolla, "Nonparametric methods for the identification of linear parameter varying systems," in *Int. Symposium on Computer-Aided Control System Design*, San Antonio, 2013.

[60] T. Johansen and B. Foss, "A NARMAX model representation for adaptive control based on local models," *Modeling, Identification and Control,* vol. 13, no. 1, p. 25–

39, 1992.

[61] T. Johansen and B. Foss, "Identification of non-linear system structure and parameters using regime decomposition," *Automatica,* vol. 31, no. 2, p. 321–326, 1995.

[62] R. Babuska and H. Verbruggen, "Neuro-fuzzy methods for nonlinear system identification," *Annual Reviews in Control,* vol. 27, no. 1, p. 73–85, 2003.

[63] T. Johansen and B. Foss, "Operating regime based process modeling and identification," *Computers and Chemical Engineering,* vol. 21, no. 2, p. 159–176, 1997.

[64] S. A. Billings, "Models for Linear and Nonlinear Systems," in *Nonlinear System Identification*, New York, Wiley, 2013, p. 607.

[65] G. Gregorčič and G. Lightbody, "Nonlinear system identification: From multiple-model networks to Gaussian processes," *Engineering Applications of Artificial Intelligence,* vol. 21, no. 7, p. 1035–1055, 2008.

[66] Q. Wei-min and J. Xiao, "Fuzzy Predictive Control for a Tubular Heat Exchanger System Based Multiple Models Strategy," in *5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, 2013.

[67] N. Daroogheh, "Centrifugal compressor identification using LOLIMOT," in *Chinese Control and Decision Conference*, Guilin, 2009.

[68] A. Wolfram, D. Fussel, T. Brune and R. Isermann, "Component-based multi-model approach for fault detection and diagnosis of a centrifugal pump," in *Proceedings of the 2001 American Control Conference*, Arlington, 2001.

[69] O. Galan, J. A. Romagnoli and A. Palazoglu, "Real-time implementation of multi-linear model-based control strategies: An application to a bench-scale ph neutralization reactor," *Journal of Process Control,* vol. 14, no. 5, pp. 571-579, 2004.

[70] T. V. Costa, A. M. F. Fileti, L. C. Oliveira-Lopes and F. V. Silva, "Experimental assessment and design of multiple model predictive control based on local model networks for industrial processes," *Evolving Systems,* vol. 6, no. 4, pp. 243-253, 2015.

[71] J. Novák, P. Chalupa and V. Bobál, "MIMO model predictive control with local linear models," in *Proceedings of the 13th WSEAS international conference on*

Automatic control, modelling & simulation, Stevens Point, 2011.

[72] J. Novák, P. Chalupa and V. Bobál, "Modelling and predictive control of a nonlinear system using local model network," in *IFAC Proceedings Volumes* , Milano, 2011.

[73] J. Ragot, "Diagnosis and control using multiple models: Application to a biological reactor," in *nternational Symposium on Advanced Control of Industrial Processes*, Hangzhou, 2011.

[74] M. Mola, M. A. Khanesar and M. Teshnehlab, "Subspace identification of dynamical neurofuzzy system using LOLIMOT," in *IEEE International Conference on Systems, Man and Cybernetics*, Istanbul, 2010.

[75] S. C. McLoone and G. W. Irwin, "On nonlinear modelling using velocity-based multiple model networks," in *Proceedings of the 2001 American Control Conference*, Arlington, 2001.

[76] D. J. Leith and W. E. Leithead, "Analytic framework for blended multiple model systems using linear local models," *International Journal Of Control,* vol. 72, no. 7, p. 605–619, 1999.

[77] T. A. Johansen and B. Foss, "Constructing NARMAX models using ARMAX models," *International Journal of Control,* vol. 58, no. 5, pp. 1125-1153, 1993.

[78] S. A. Billings and W. S. F. Voon, "Piecewise linear identification of non-linear systems," *International Journal of Control,* vol. 46, no. 1, p. 215–235, 1987.

[79] R. Orjuela, B. Marx, J. Ragot and D. Maquin, "Nonlinear system identification using heterogeneous multiple models," *International Journal of Applied Mathematics and Computer Science,* vol. 23, no. 1, p. 103–115, 2013.

[80] S. El Ferik, "Modeling and Identification of Nonlinear Systems: A Review of the Multimodel Approach - Part 2," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. PP, no. 99, pp. 1-9, 2016.

[81] R. Murray-Smith and T. A. Johansen, "Local learning in local model networks," *IEE Conference Publication,* no. 409, pp. 40-46, 1995.

[82] JModelica.org, "User Guide version 1.3.0," 2010.

[83] R. De Coninck, "Toolbox for development and validation of grey-box building models for forecasting and control," *Journal of building performance simulation,*

vol. 9, no. 3, pp. 288-303, 2016.

[84] B. Adams, L. Bauman, W. Bohnhoff, K. Dalbey, M. Ebeida, J. Eddy, M. Eldred, P. Hough, K. Hu, J. Jakeman, J. Stephens, L. Swiler, D. Vigil and T. Wildey, "Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 User's Manual," Sandia Technical Report SAND2014-4633, 2014.

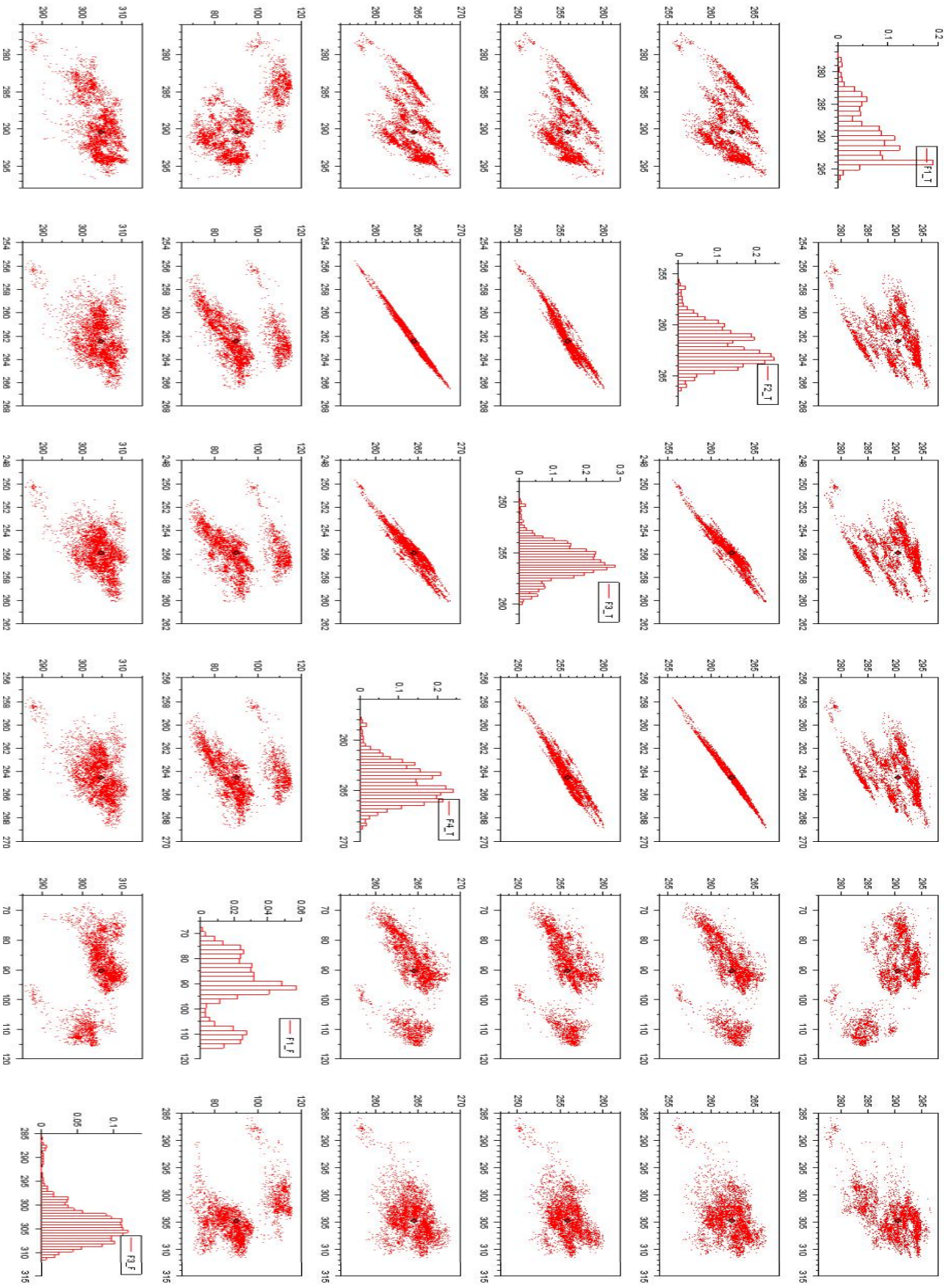
[85] A. Lovasz, P. Mizsey and Z. Fonyo, "Methodology for parameter estimation of modelling of pervaporation in flowsheeting environment," *Chemical Engineering Journal,* vol. 133, no. 1, pp. 219-227, 2007.

[86] ProSim, "www.prosim.net," [Online]. Available: http://www.prosim.net/bibliotheque/File/Brochures/Brochure_PSP-En_compressed.pdf. [Accessed 16 4 2017].

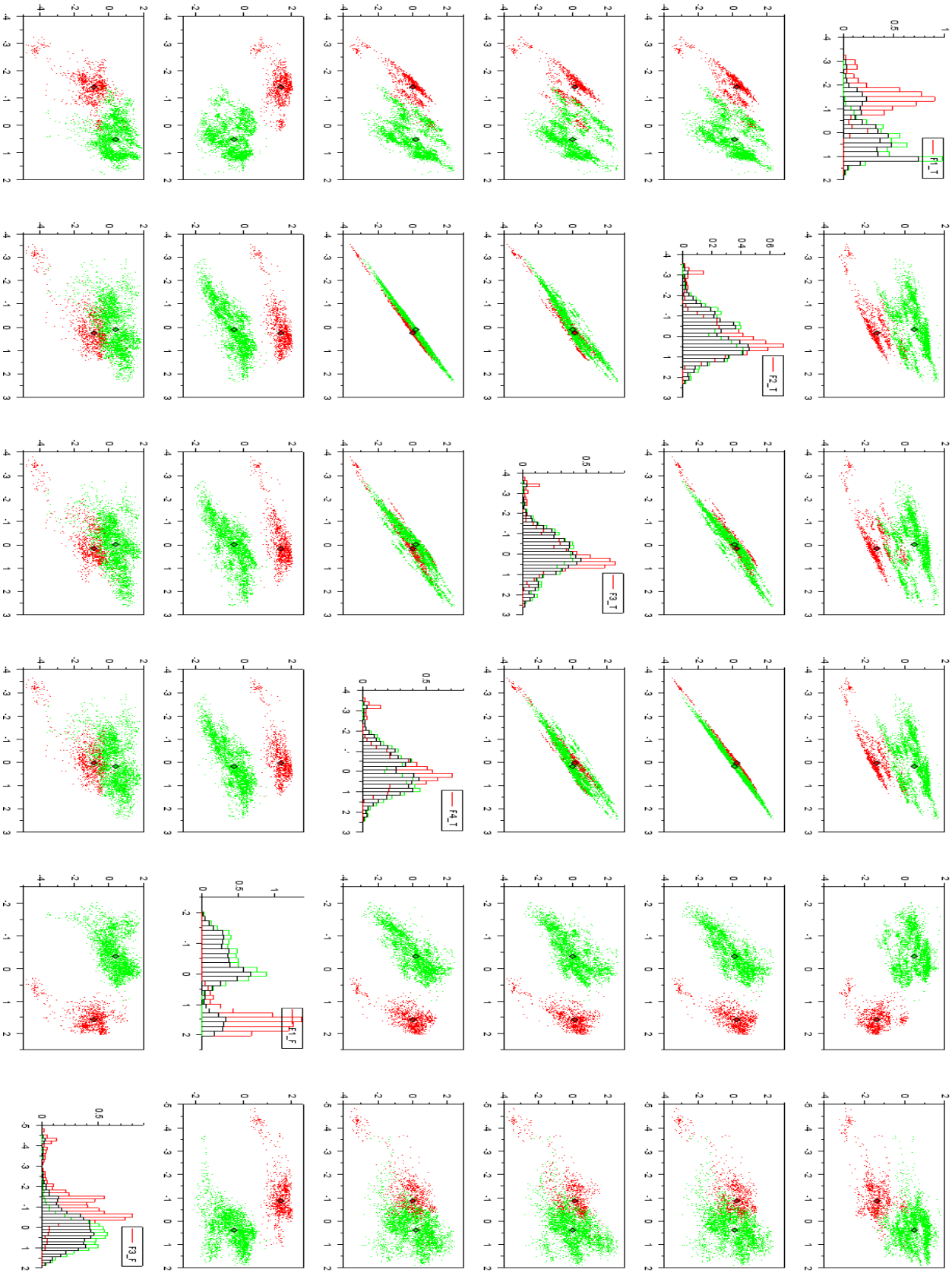
[87] ProSim, "Operating balance optimization of a natural gaz liquids plant," ProSim, 2016.

[88] N. H. Saeid and K. N. Seetharamu, "Finite element analysis for co-current and counter-current parallel flow three-fluid heat exchanger," *International Journal of Numerical Methods for Heat and Fluid Flow,* vol. 16, no. 3, pp. 324-337, 2006.

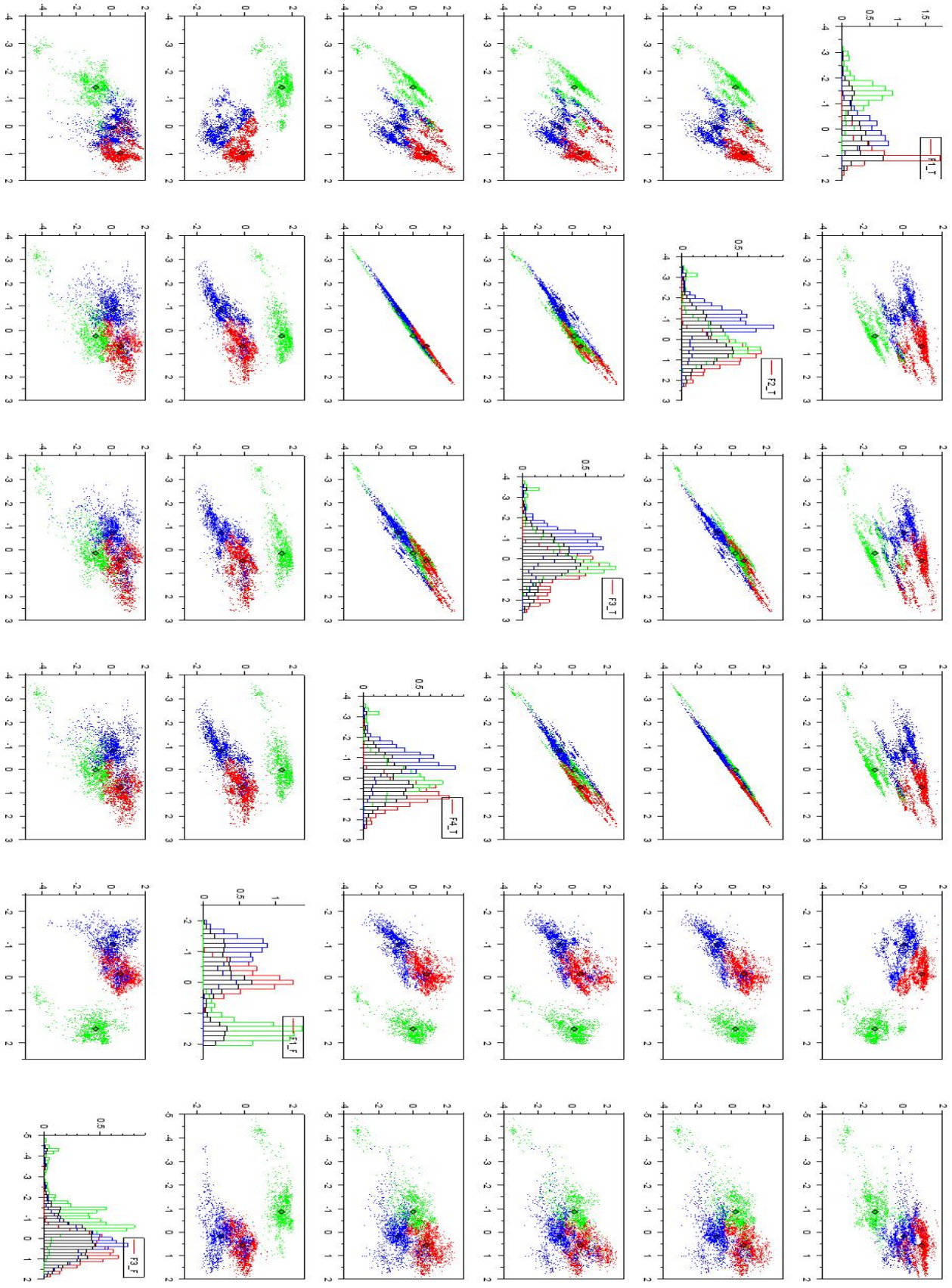
Appendix 1 - Scatter matrix



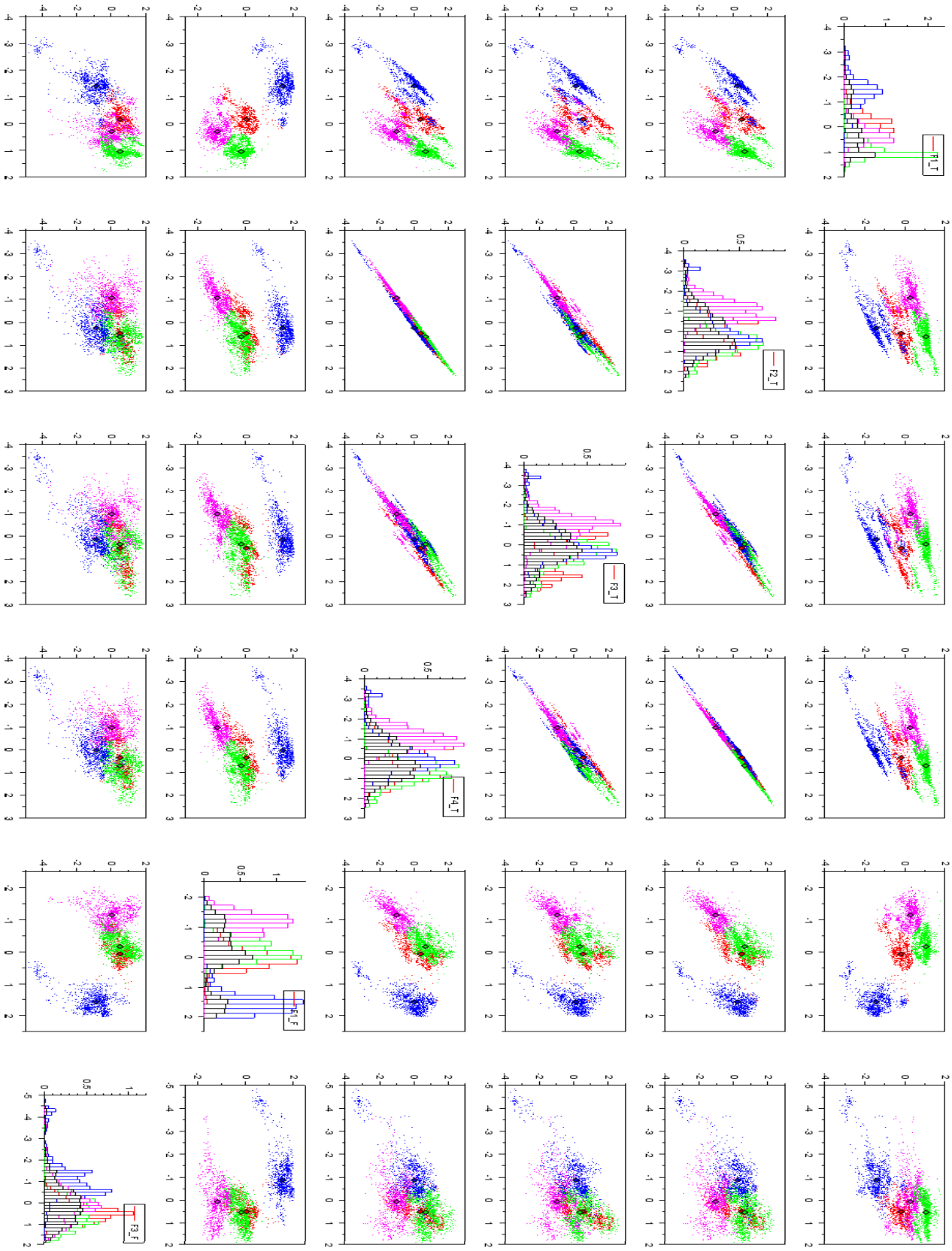
Appendix 2 - K-means++, 2 clusters



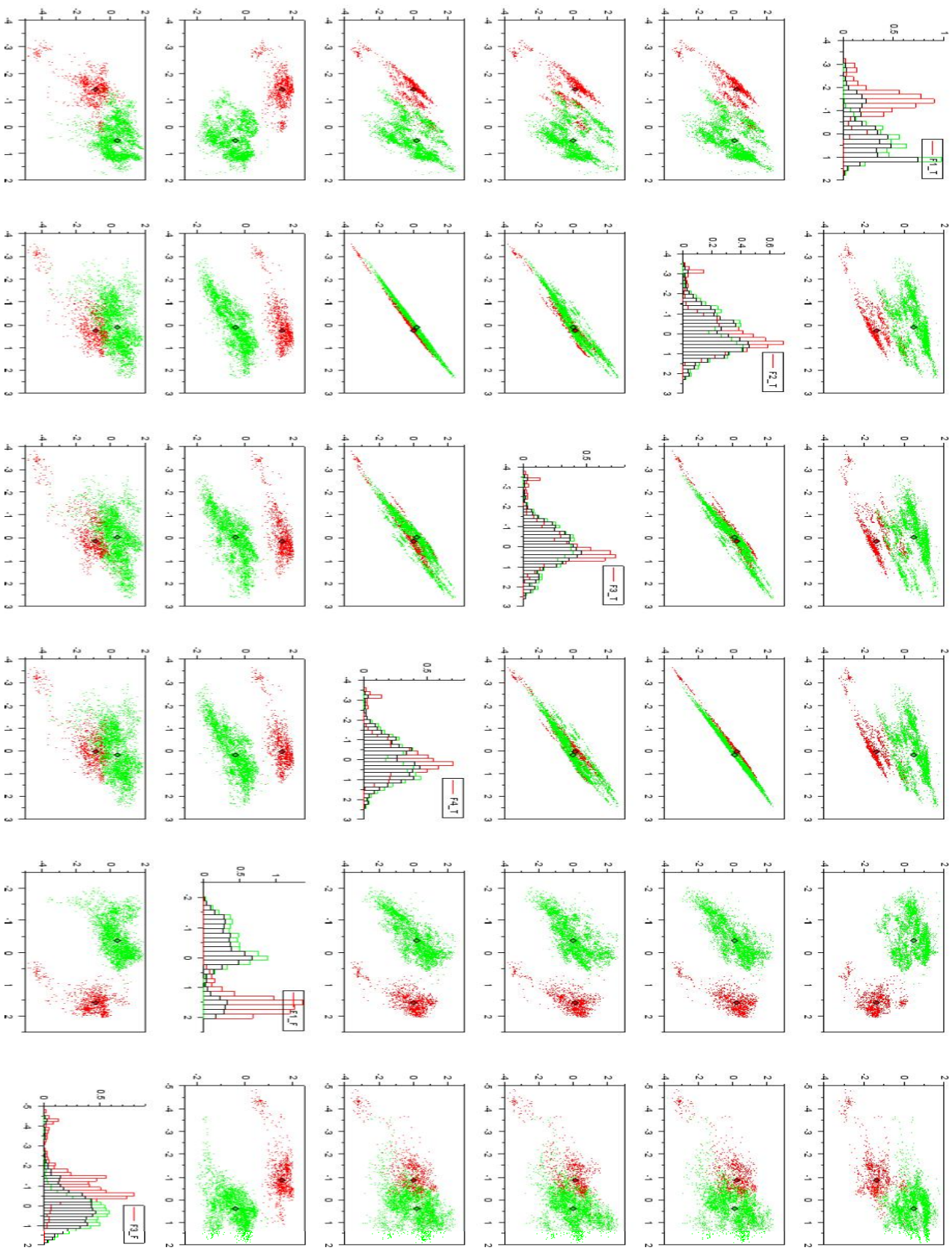
Appendix 2 - K-means++, 3 clusters



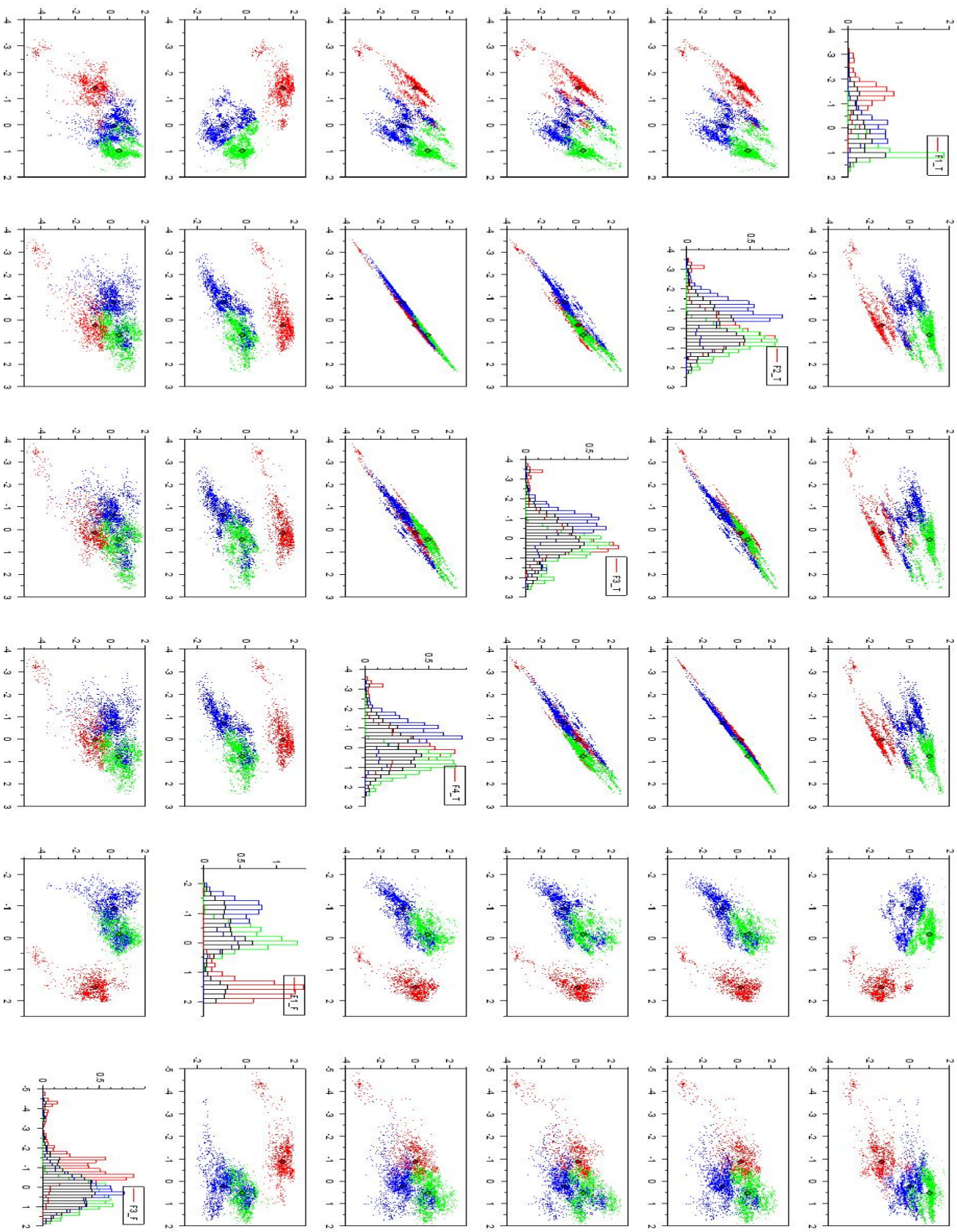
Appendix 2 - K-means++, 4 clusters



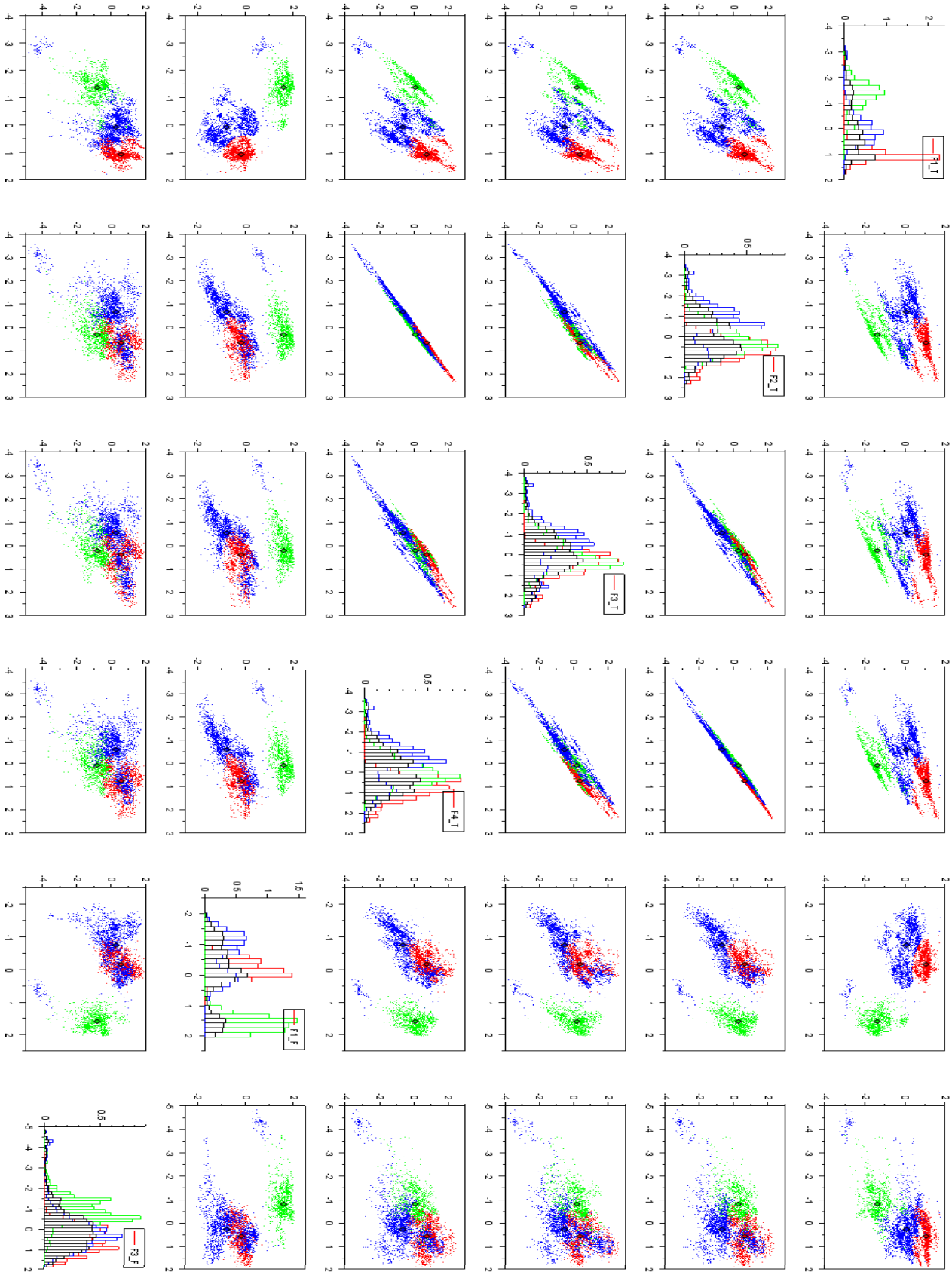
Appendix 2 - Kernel K-means, 2 clusters and fhyper 0.1



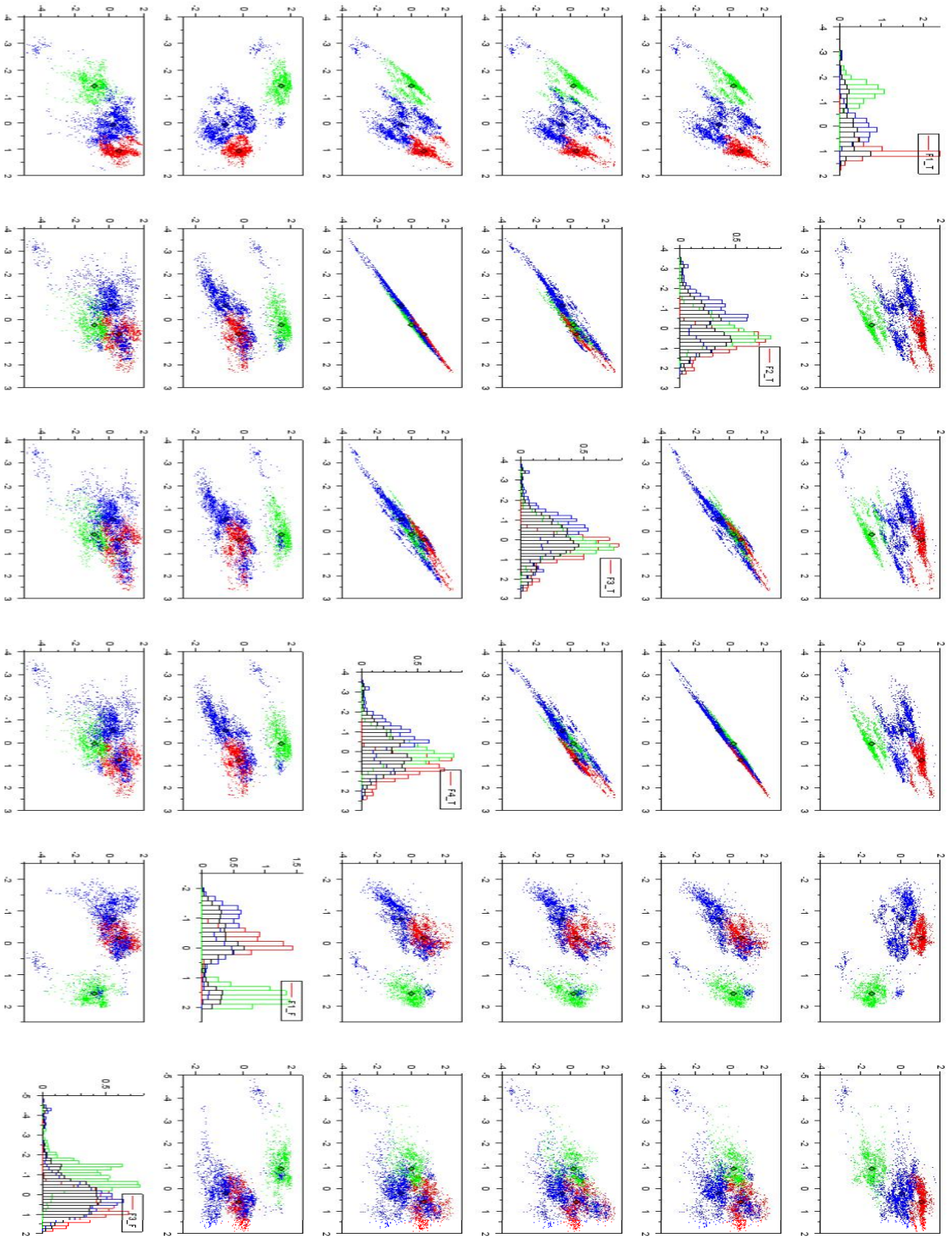
Appendix 2 - Kernel K-means, 3 clusters and fhyper 0.1



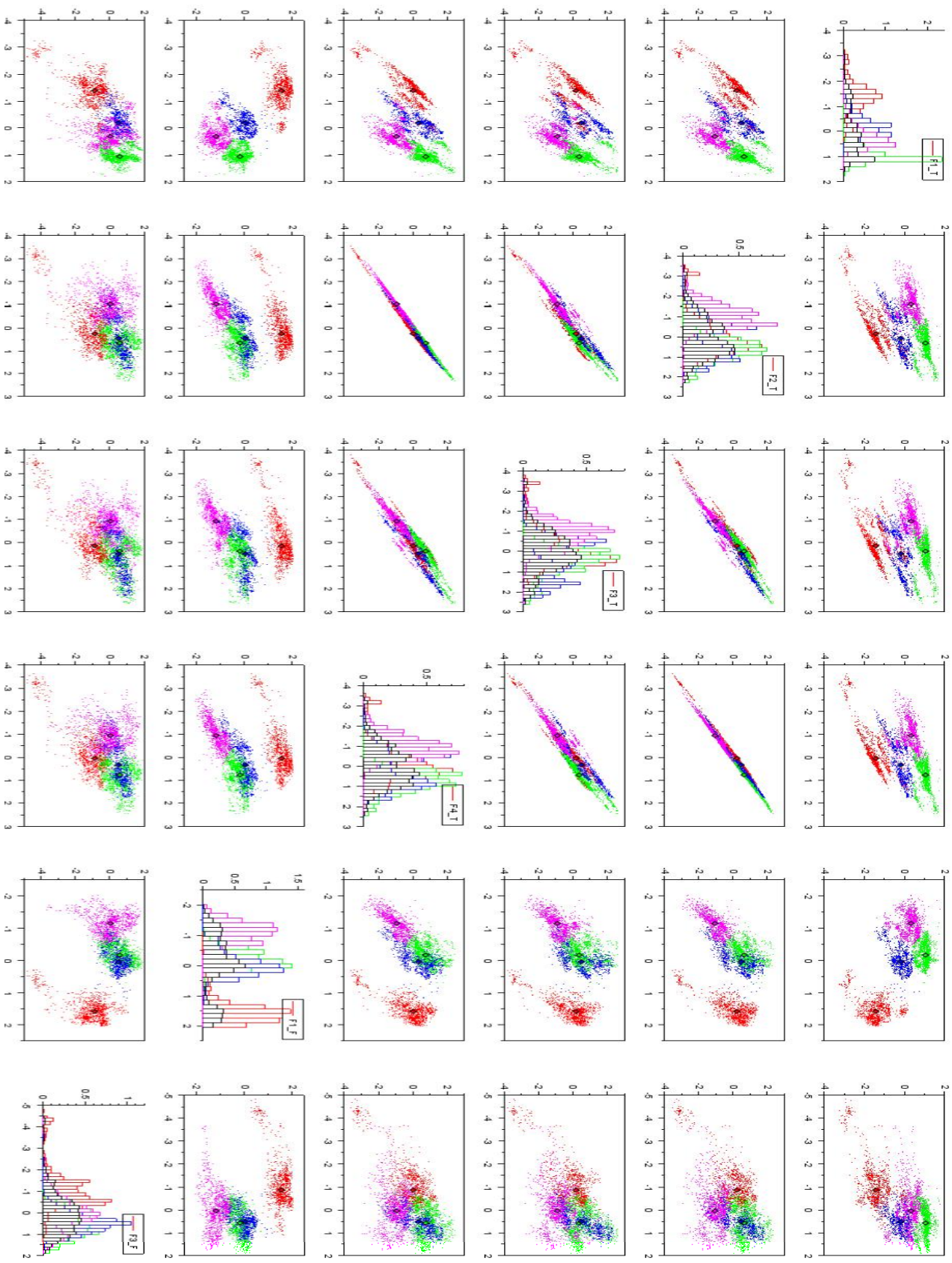
Appendix 2 - Kernel K-means, 3 clusters and fhyper 1



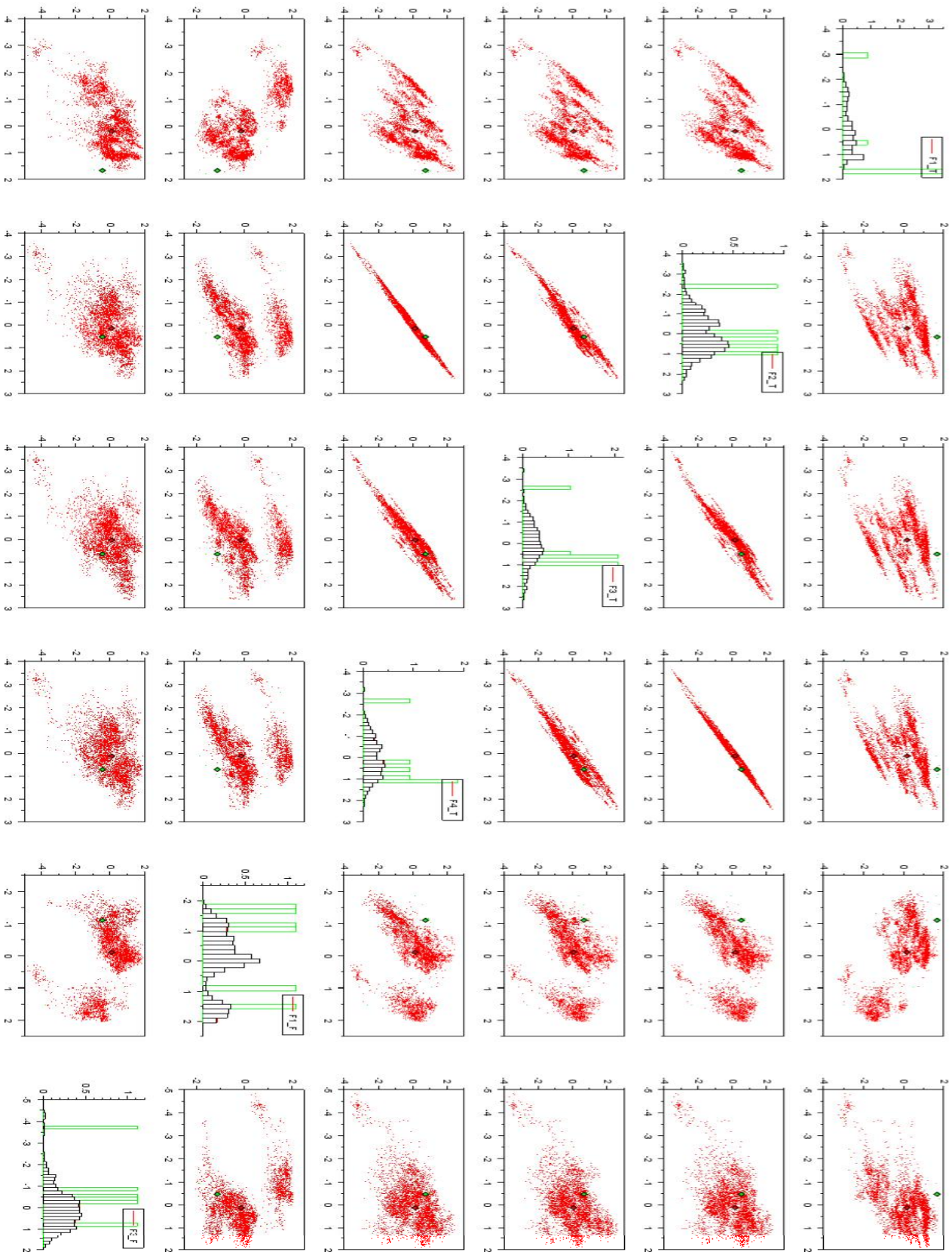
Appendix 2 - Kernel K-means, 3 clusters and fhyper 2



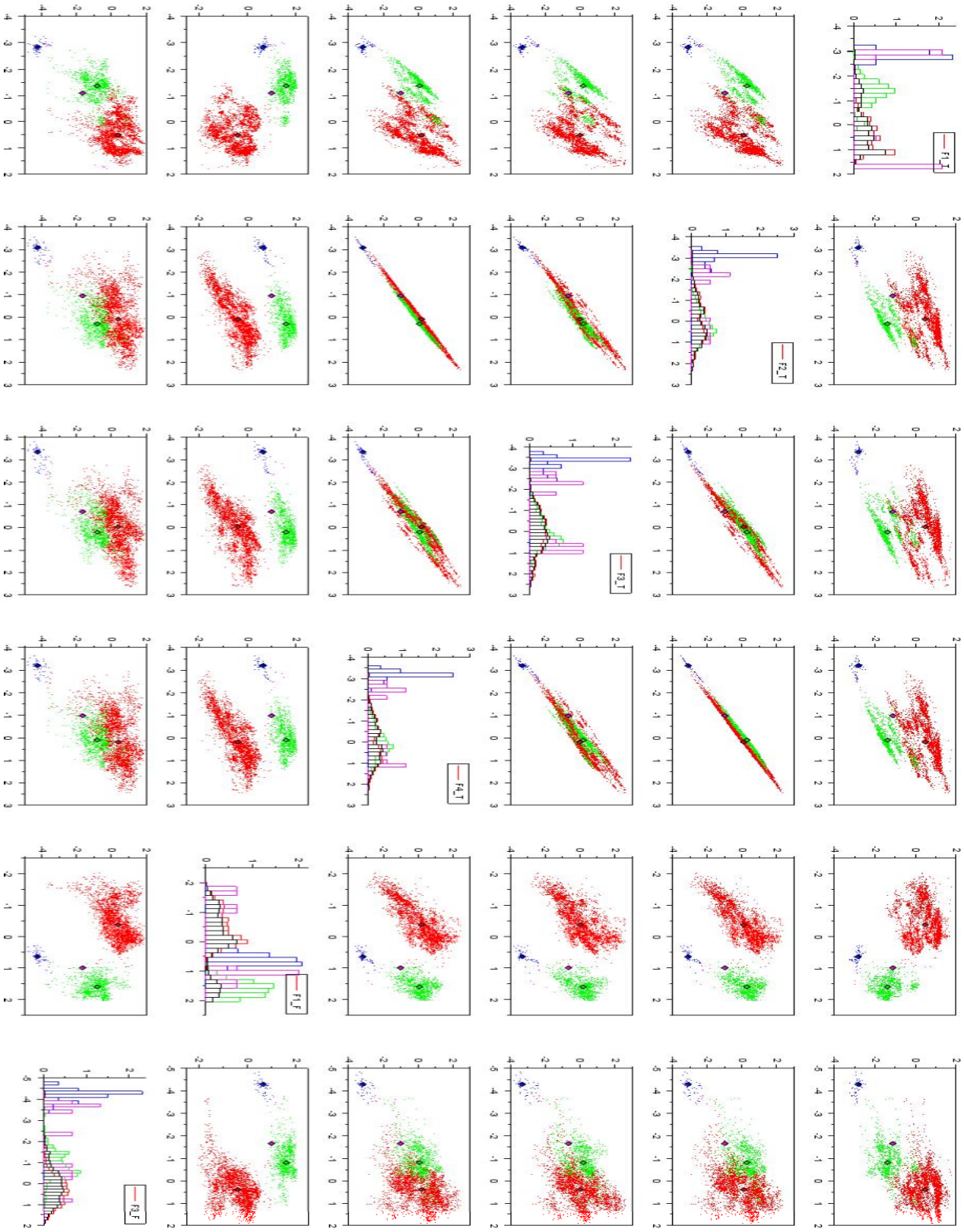
Appendix 2 - Kernel k-means, 4 clusters and fhyper 1



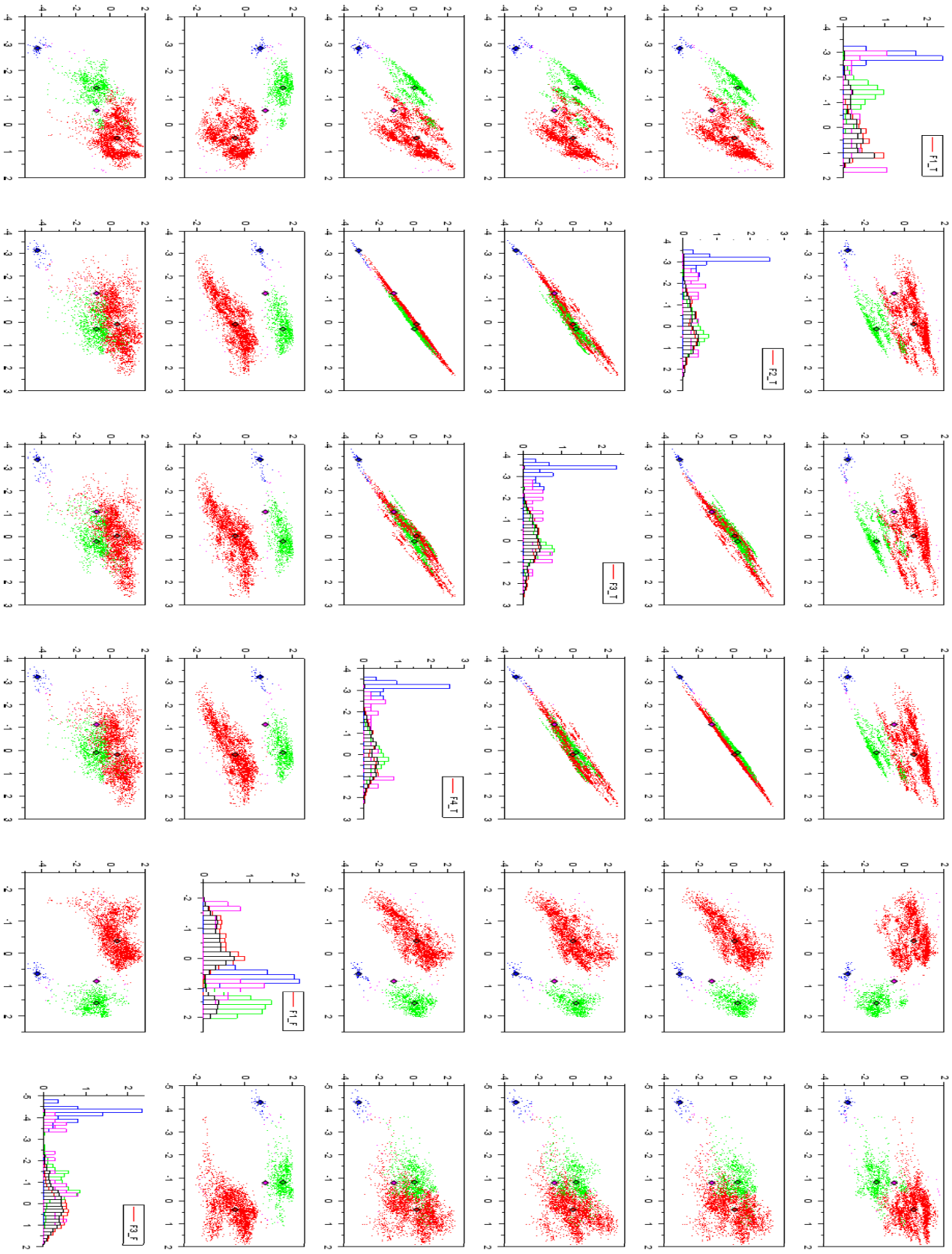
Appendix 2 - DBSCAN, ε = 0.2, minPts = 30



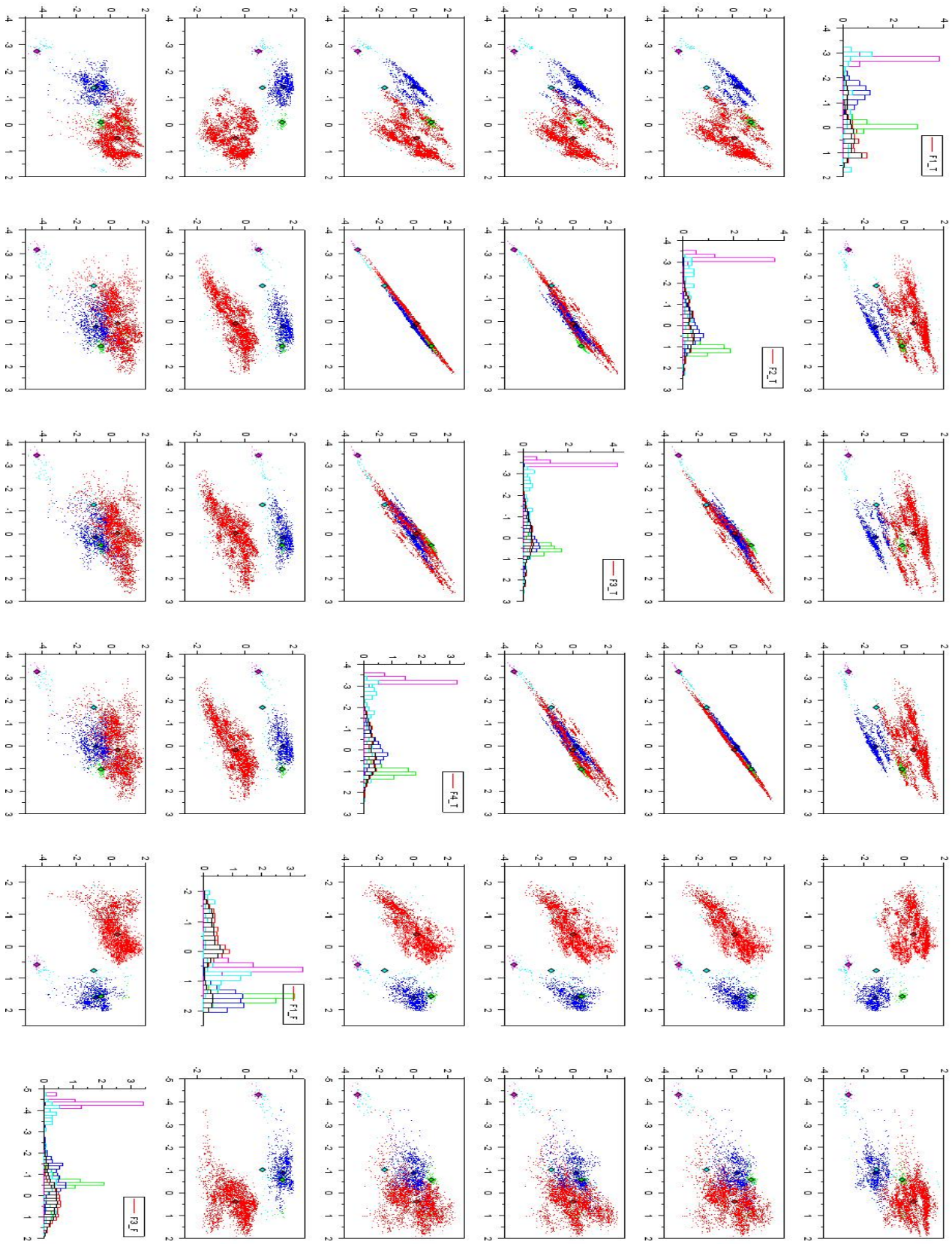
Appendix 2 - DBSCAN, ε = 0.15, minPts = 30



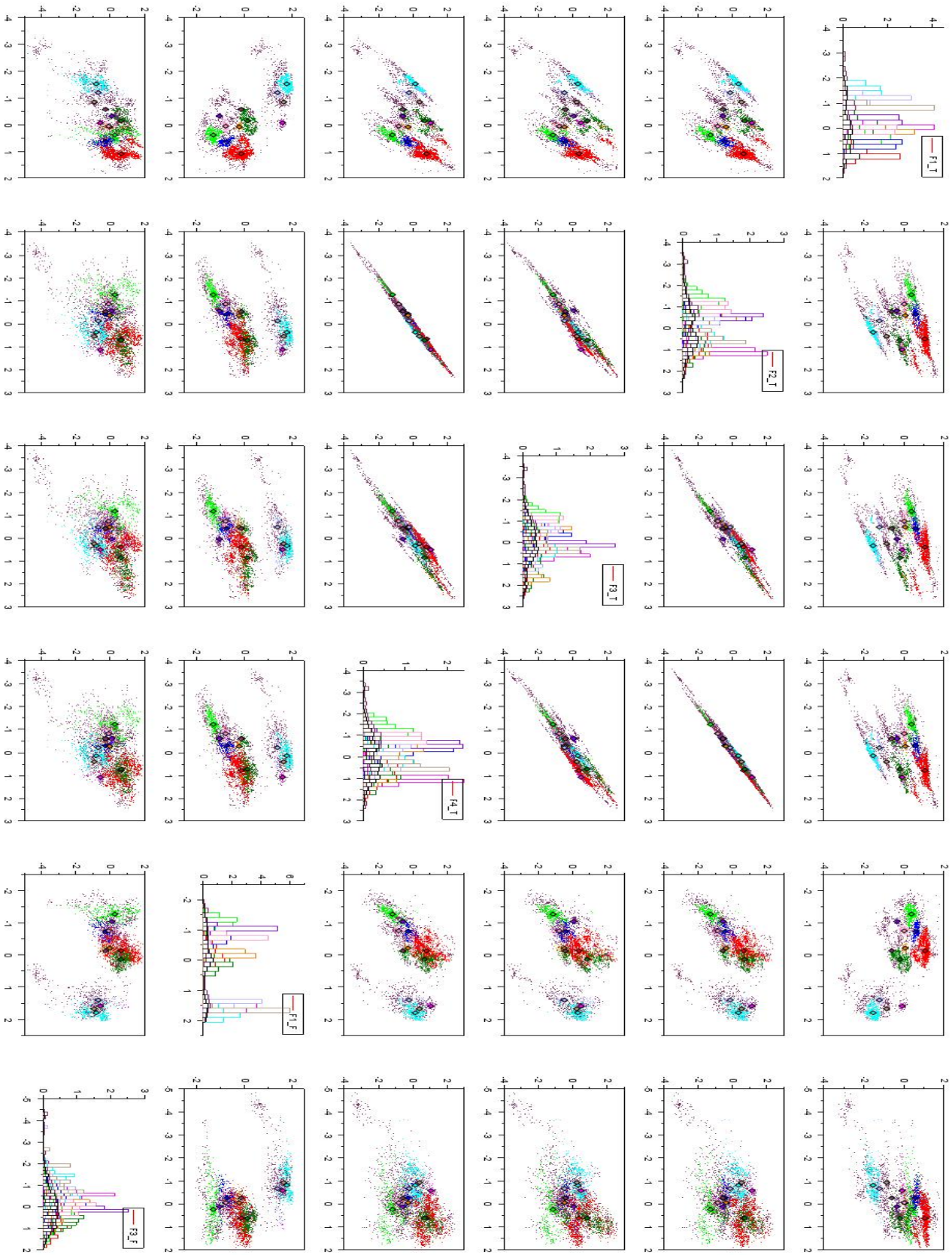
Appendix 2 - DBSCAN, ε = 0.1, minPts = 30



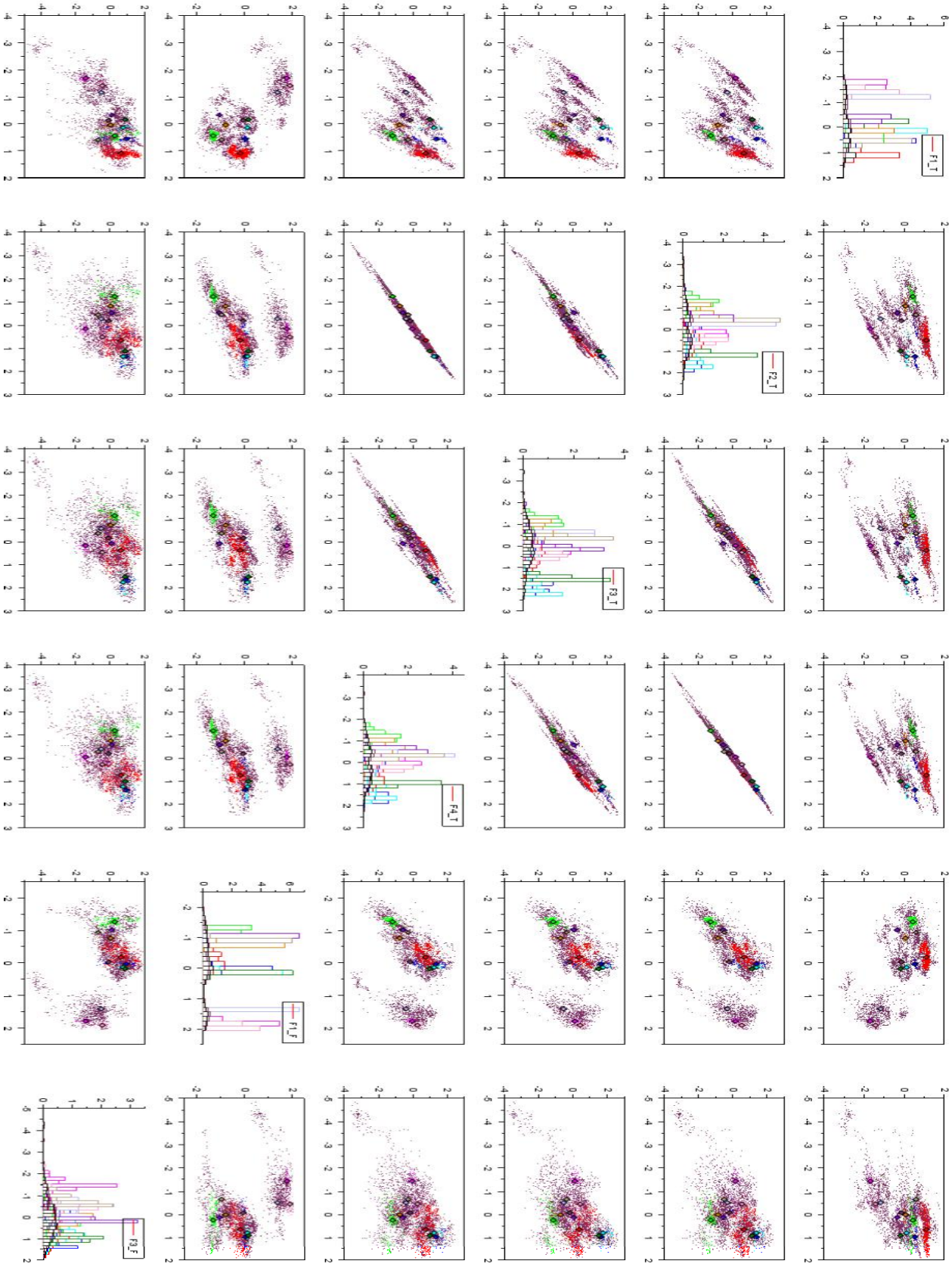
Appendix 2 - DBSCAN, ε = 0.05, minPts = 30



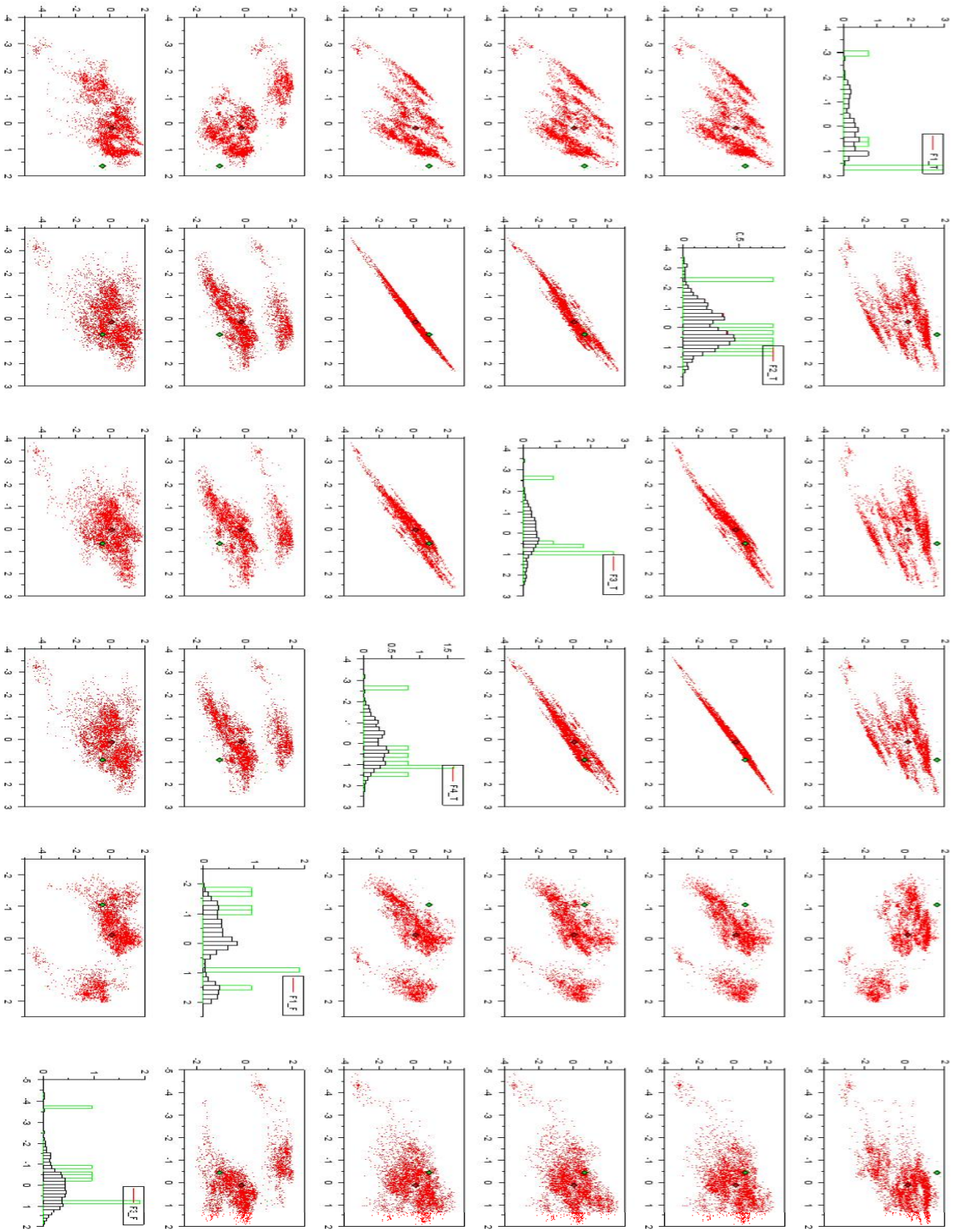
Appendix 2 - DBSCAN, ε = 0.01, minPts = 30



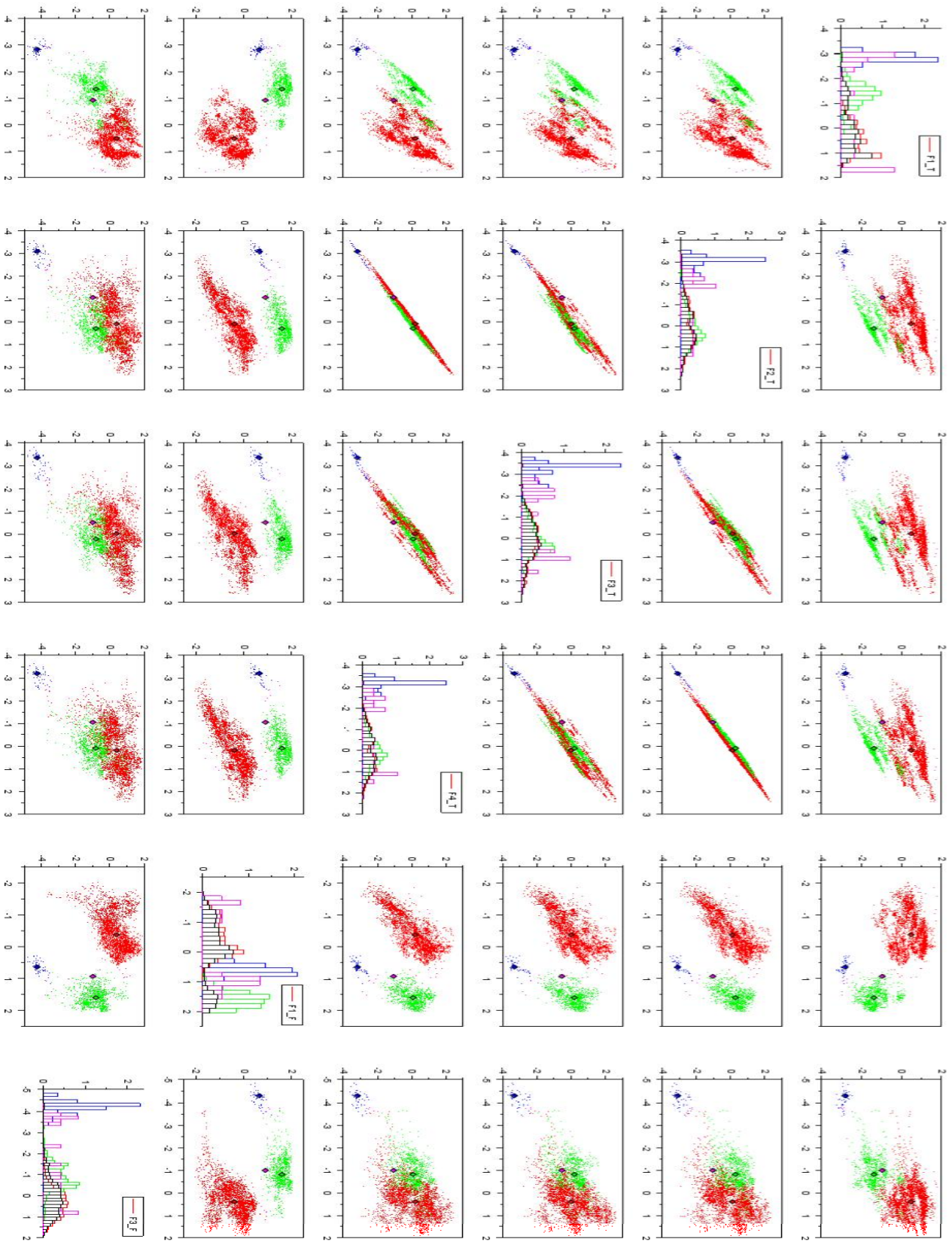
Appendix 2 - DBSCAN, ε = 0.005, minPts = 30



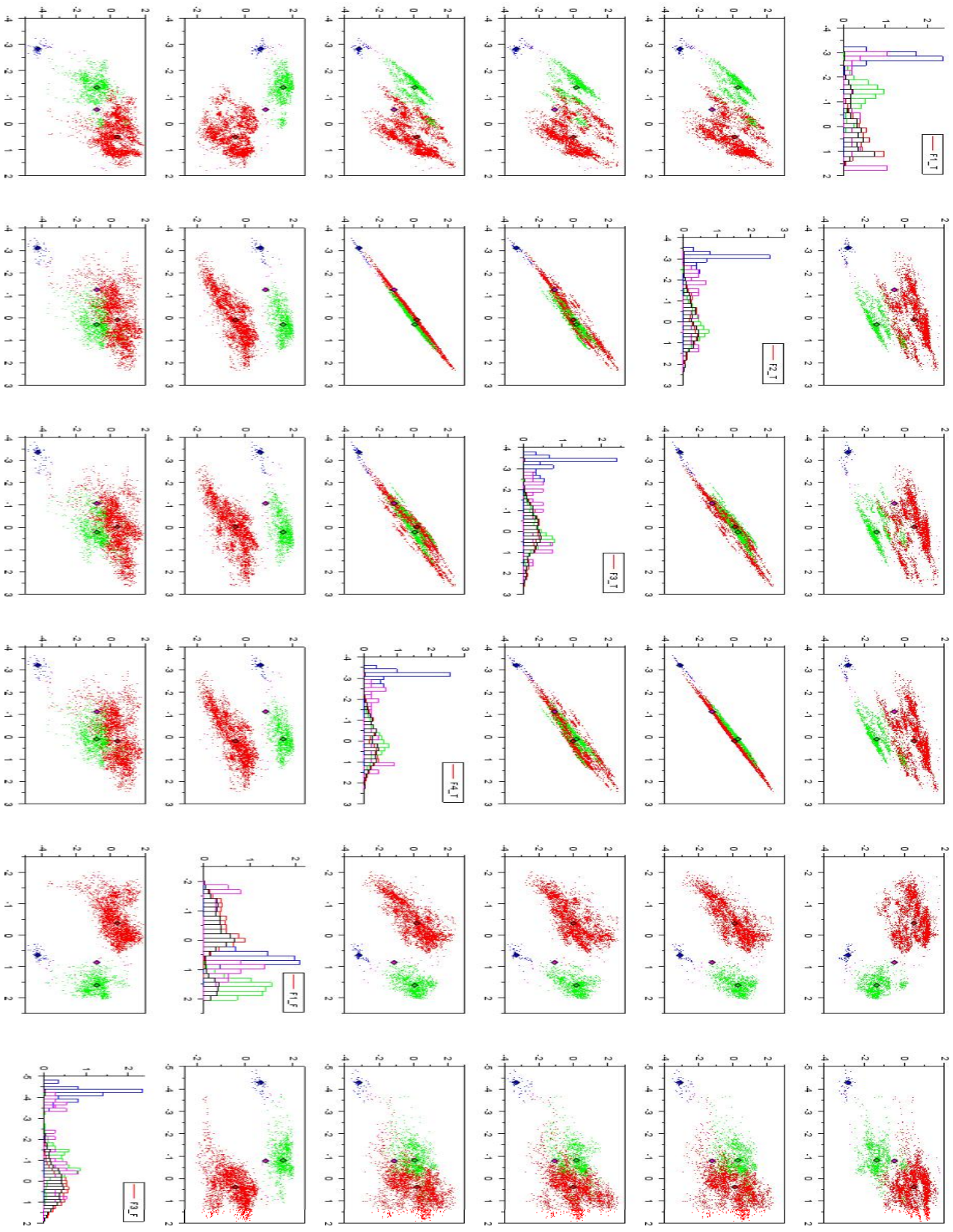
Appendix 2 - DBSCAN, ε = 0.1, minPts = 10



Appendix 2 - DBSCAN, ε = 0.1, minPts = 20



Appendix 2 - DBSCAN, ε = 0.1, minPts = 30



Appendix 2 - DBSCAN, ε = 0.1, minPts = 40